Author:
Barker, Tim

Title:
Design and Numerical Validation of Decentralised Algorithms for the Control of Road Networks

Design and Numerical Validation of Decentralised Algorithms for the Control of Road Networks

T. Barker
Department of Engineering Mathematics
University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

October 2017

Word count: Thirty-Three Thousand Four Hundred

Abstract

Congestion on road traffic networks is a problem. Control theory can be applied to reduce congestion, and new technologies present more opportunities to do this. Strategies can be centralised or decentralised, but decentralised strategies offer advantages in terms of feasibility and scalability. We propose two decentralised control algorithms to be applied to road networks, controlling both vehicle routes and traffic lights. We validate these algorithms numerically using a microscopic traffic simulator.

We introduce the current literature on vehicle routing, and intersection control, providing an overview of each. We present a decentralised routing methodology, by which vehicles pick their routes by minimising a cost function based on travel time and road occupancy. We investigate the effect of tuning the control parameter which determines the relative balance between the two components of the cost function, and find that this is dependent on network topology and the presence of traffic lights. We find the algorithm performs favourably compared to the shortest path, and contrast our algorithm with Dynamic User Assignment using Gawron's iterative approach, showing the proposed method has distinct advantages in specific network topologies. We prove that in a scenario where all other vehicles are routed using only the shortest path, only a fraction of the vehicles in the network are required to adopt our proposed routing strategy to make a significant reduction in delays.

We present a modular decentralized traffic light controller and propose several algorithms which harness the potential of Vehicle-to -Infrastructure communication. We compare the performance of these algorithms with those already found in the literature, as well as exploring the impact of network topology on the performance of the controller. We find that our algorithms are able to outperform other controllers, but there is a significant relationship between network topology and algorithm performance. We then test these algorithms in the Luxembourg network and find the improvements in travel-time carry over to a real world scenario.

Dedication and acknowledgements

On the path to completing this thesis, I have received a great deal of time and expertise from a number of people, without whom it may never have been completed. I would especially like to thank my supervisor Professor Mario Di Bernardo, who has been a great mentor to me and guided me through the last 3.5 years with a great deal of encouragement, teaching and patience. I would also like to thank Dr Chao Zhai, Dr Gianfranco Fiore and Dr Giovanni Russo for all of their help in completing various parts of my research. I have been incredibly fortunate in having the chance to work with each of you and have benefited a great deal from your experience. I would like to thank Dr Filippo Simini for the time he spent reviewing me each year, and the detailed comments he gave me on my written work.

I would like to thank those academics who hosted me at their labs during my PhD: Professor Carlos Canudas-De-Wit and his team at INRIA, Grenoble, and Dr Simon Box who was with the University of Southampton. These experiences were both informative and enjoyable and gave me the insights necessary to go on to improve my research and eventual dissertation.

Finally, I would like to thank my family, who supported me fully when I told them that I wanted to return to University for another 3 years to achieve a PhD and have been a rock throughout my time doing it, and Mariya, who has been and always will be my biggest cheerleader. This dissertation is dedicated to you.

This thesis would never have happened in any form without the support of the James Dyson Foundation. It is a testament to James Dyson that he supports engineering education, and my heartfelt appreciation goes to all those at the University of Bristol and Dyson who made it happen.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE:.....

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Transportation is of fundamental importance to a high functioning economy. The rapid and efficient movement of goods and people has enabled commerce on a massive scale. However, densely populated urban areas are plagued by transport networks which operate below their maximum capacity due to congestion.

TomTom International, a company famous for their Global Positioning System (GPS) devices, produce a congestion index for over 180 cities across the world [102,103]. TomTom express congestion as the percentage difference in travel times between peak and off-peak periods of road usage. According to TomTom's research cities across the UK suffer from delays due to congestion during peak periods, including Birmingham with a congestion index of 23%, and London with a congestion index of 34%.

A report for INRIX, a transportation analytics company, in 2014 found that population growth and increasing GDP per capita would drive up demand for road transport in the world's largest economies. Their findings suggest that the costs of traffic for the UK, Germany, France and the US alone will rise 46% by the year 2030. The cumulative cost across all 4 nations, accounting for time spent by citizens in traffic, indirect costs such as the costs due to pollution and other direct costs such as the cost of gas, will have reached $4.4 trillion by 2030 [18].

There are a number of factors that may contribute to congestion on a traffic network. An obvious problem is the number of vehicles using the network [31]. A network with a fixed capacity for vehicles will undoubtedly suffer congestion if the traffic on it exceeds that capacity, for example during peak periods. Furthermore, if roads become congested then the congestion will tend to spread to other roads in the network. If techniques can be implemented to avoid congestion in the first place, or to clear congested roads faster, then the whole network can benefit.

Solving the problem is difficult. Building new roads or widening the existing ones does not necessarily reduce congestion, as it can encourage more people to drive their car, and the behaviour of drivers tends to lead to inefficient use of the road network anyway. [17,26]. Governments can intervene by encouraging the use of other modes of transport, such as buses, cycling and walking, and technology can ultimately make all of these transport

options better and more efficient.

Taking a network perspective of the road layout, with junctions represented by nodes and roads represented by edges, then congestion is likely to occur around any node with a high centrality[1], as these nodes are likely to appear on the shortest path for many vehicles [122]. This creates bottleneck situations at certain points in the traffic network and may leave other parts of the traffic network underused. If access to the nodes with high centrality is controlled by a traffic light, then the performance of the traffic network will further depend on whether the timing of the traffic light is optimal.

Car drivers themselves are autonomous agents who make independent decisions about their driving actions, such as the route they take. Whilst it is possible to use historical data and other information in order to predict traffic movements, it is not possible to plan for the movements and positions of all drivers in advance. Where possible drivers should be influenced to behave in a way that benefits the network as a whole, such as avoiding these high centrality nodes.

In general, the problem is that of devising network management strategies able to maximise (or minimise) both system and user control objectives. The system objectives are those relating to the performance of the entire network, such as the average delay experienced by all drivers. The user objectives are those relating to the experience of an individual driver, such as their actual delay time.

Upon first consideration, the system objectives would seem to be aligned with those of the user, for example from both the system perspective and the user perspective it is advantageous for the user to leave the network as quickly as possible. However, despite this mutual objective, the selfish nature of individual users (such as picking the fastest route, or not wanting to wait at traffic lights) can adversely affect the system, and in turn, make the user's journey worse. An example of this behaviour is provided in Section 2.3.1, where we review a simple well-known case of route choice behaviour, which shows that when users are given free choice, the decision to speed up their own journeys can result in slower journeys for all users, including themselves. The example illustrates the simple fact that from a system perspective there is more information available to make the best decision, whereas for a single user their desire to improve their current situation may actually make it worse because that decision is not made in isolation.

Below, the key objectives are listed of the network management strategies we wish to develop in this thesis:

1. Minimise the average delay, defined as the difference between a traveller's actual travel time, and the minimum possible travel time they could have expected.

2. Minimise the standard deviation in delay for all travellers. This means that all travellers experience approximately equal delays if delays are occurring.

3. Maximise predictability in travel times and delays, making journey time estimates highly reliable. This includes daily variations during the day, such as the morning

[1]Centrality in this sense refers to the nodes which are on the fastest path between many other nodes, and so are associated with high flows through them.

and afternoon peaks due to working hours, and also weekly, monthly and yearly variations due to the sensitivity of traffic to small changes.

These objectives are interdependent, and clearly subject to a notable trade-off. It may be possible to create strategies that create a lower average delay but unfairly penalise certain drivers. It may also be possible to come up with very good strategies for unsaturated traffic conditions, but that perform wildly differently under saturated traffic conditions.

Addressing these issues to alleviate some of the problems of modern transport is the goal of the emerging research field known as Intelligent Transportation Systems (ITS) [95]. Within ITS there are many areas, some looking at specific transport solutions and others looking at integrated transportation systems (e.g. rail, air, road).

In general, current research in ITS can be broken down into five main areas [85] :

1. Design of advanced traveller information systems (for example, equipping drivers with real-time traffic information).

2. Development of advanced transportation management systems (such as traffic signals and traffic operations centres).

3. Synthesis of ITS-enabled transportation pricing systems (which are primarily focused on optimal congestion pricing).

4. Design of advanced public transportation systems (including the provision of information such as real-time location of public transport to passengers).

5. Implementation of fully integrated intelligent transportation systems (covering areas such as vehicle-to-infrastructure and vehicle-to-vehicle communication).

The research work presented in this thesis is aimed at developing better management systems for road networks and therefore falls in the second area listed above. Control theory can offer methods and techniques suitable for addressing these issues. There is a deficit of control actions that can be directly taken on a traffic network, where much depends on the movement and behaviour of free, indestructible agents, whose behaviour cannot be predicted with 100% certainty. However, new technologies, such as observers for real-time traffic speed, Vehicle-to-Infrastructure (V2I) communication and autonomous vehicles, are improving both the ability to observe the state of the network and exert possible control actions and so new research must take advantage of these changes in technology and introduce methods which take advantage of them.

1.2 Aims and Hypothesis

The aim of the thesis is to explore applications of control theory to road traffic control in the areas of vehicle routing and intersection control. Specifically, we develop new control architectures in these areas and algorithms to improve congestion across various road network topologies.

Road networks are multi-variate time-varying systems. The scale of road networks makes the problem of optimisation intractable, and even where centralised control has been implemented it must operate using heuristics or settle for suboptimal strategies. The key hypothesis in this dissertation is that decentralised control can result in behaviour that yields high performance with low computational and communication overheads. Such decentralised algorithms do not require a centralised controller, scale with $\mathcal{O}(1)$ complexity, and are robust to disturbance or changing demands.

1.2.1 Contribution and Deliverables of the Thesis

This thesis contributes two decentralised control frameworks aimed at reducing congestion in road networks via the implementation of:

1. Decentralised vehicle routing
2. Decentralised intersection control

Algorithms have been developed within each of these frameworks and then extensively tested via numerical validation across various network topologies. These algorithms have been shown to increase the resilience of the networks to congestion when compared to a naive alternative such as shortest path routing for fixed-cycle control. Furthermore, it was demonstrated that network topology is a significant factor in the performance of the algorithms. The algorithms draw from existing knowledge in the literature (explored in Chapters 2 and 3), but the specific details of their implementation are notable for being, as far as the authors have been able to ascertain, completely original in their approach and validation across multiple network topologies. Notably the some of the results pertaining to the development of intersection control algorithms in Chapters 5 and 6 in this thesis have been published in the conference proceedings for the '5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems' [9]. A significant amount of software development has been done in order to allow the numerical validation of these algorithms. In order to obtain the results presented in this thesis it was necessary to extend open-source traffic modelling software with custom libraries to both test the algorithms and present the data from simulations.

1.3 Thesis Outline

The rest of the Thesis is outlined as follows:

Chapter 2 provides a background to the problem of vehicle routing on a network. The problem of finding a path between two points in any network is first covered, followed by how these networks can be modelled specifically as traffic networks. We then cover traffic assignment problems on these networks. An overview of the proposed centralised and decentralised approaches, their advantages and their limitations is provided.

Chapter 3 provides a background to the problem of intersection control in a network. Intersection control has been of interest since traffic lights were first introduced. The

initial problem was to model delay to vehicles at a single intersection and attempt to minimise it by careful selection of the controllable parameters (such as the cycle time). Approaches have been extended in order to coordinate networks of intersections and maximise throughput, but they have limitations in terms of computation time and effectiveness. These approaches are explored, and decentralised alternatives are discussed, specifically, an overview of approaches based on the principle of work-conservation is presented, which offer a compelling design strategy for decentralised control of intersections.

Chapter 4 covers the design and implementation of a new Decentralised Routing Algorithm (DRA). Using a combination of expected travel time and local road occupancy we explore if a simple cost function can be used for decentralised decision making, which results in a lower mean delay to vehicles in the network. The algorithm is shown to balance loads on synthetic networks which results in a lower mean delay when the load on the network reaches a certain threshold value when compared with vehicles routing using only travel time.

Chapter 5 then presents the design and implementation of a new modular intersection controller. The intersection controller is modular in the sense that algorithms can be chosen independently for stage selection and stage duration. Several algorithms are developed based on the principle of work conservation.

In Chapter 6 the intersection control algorithms described in Chapter 5 are extensively tested in several synthetic networks. The algorithms are compared across several network topologies and compared against a fixed-cycle control strategy and other work-conserving controllers from the literature. We find that we are able to greatly outperform the fixed-cycle strategy.

Chapter 7 shows the application of our intersection controller to a real-world scenario in the city of Luxembourg. This scenario is based on the Luxembourg network with vehicle loading which reflects real demands over a 13-hour period. The results show that our algorithm can outperform the state-of-the-art in work-conserving controllers.

Finally, we discuss conclusions and suggestions for future work in Chapter 8.

Chapter 2

Vehicle Routing in Road Networks: An Overview

In this chapter, we will review the literature pertaining to vehicle routing. Vehicle routing touches on a broad range of topics, and a full review of the literature could constitute the work of an entire thesis in itself. Here we will not regurgitate what can be found in previous thesis on traffic networks; instead we direct the reader towards some excellent literature reviews which the authors have come across if they are interested in reviewing all areas of the topic [22, 109]. In this literature review we touch upon the topics which have been most influential in the design and validation of our decentralised routing algorithm. Firstly we review the area of graph search and optimisation. Next, we review developments in the modelling of road networks and finally study how route choice of drivers has been modelled and classified in the literature. We summarise the state-of-the-art in some key decentralised methods proposed in the literature, and then take a look at specific control theoretic approaches to routing control.

2.1 Finding a Path Through a Network

The problem of routing cars on roads has presumably been around as long as cars themselves. In the formative years of the automobile, roads were undoubtedly less busy and the route choices less numerous, and the rigorous application of mathematics was perhaps unnecessary to this early problem. As the amount of traffic on roads, and indeed the number of roads themselves, has continued to increase, vehicle routing has become a topic of much interest in various research areas (see [104, 108, 123] and references therein).

Roads are clear candidates when it comes to graph representation; little abstraction is required to envisage nodes as junctions and edges as the roads connecting them. For this reason, whilst it was not concerned with only vehicle routing in mind, Dijkstra's 1959 paper, which covered the topic of finding the shortest path between two nodes on a network, is still referenced in papers on the subject [37]. In general, graphs associated with road networks can be analysed at a ***Macroscopic*** [70] or ***Microscopic*** [78] level. Macroscopic analysis provides properties of the roads and intersections, such as the expected flow along a road based on the density of traffic. In macroscopic models, vehicles are aggregated

into flows through the network. Microscopic analysis looks at individual vehicles and requires agent-based modelling. Vehicles control their speed and route individually, and the journeys of all vehicles can be tracked explicitly.

Dijkstra's algorithm has remained the backbone of vehicle routing algorithms for over half a century, however, authors have demonstrated the benefits of heuristics in speeding up Dijkstra's discovery. These developments aim to reduce the number of nodes that must be searched, as well as the computation time required to complete them. These time savings become relevant when the networks have many nodes, or when dealing with many different search requests. The A* algorithm [51], which Hart and Nils introduced in 1968, has a specific application in graph problems with a spatial dimension, which makes it possible to focus the search for a shortest path through the network in the direction of the traveller's destination.

In the same way that Hart and Nils succeeded in speeding up Dijkstra's algorithm, so others have succeeded in manipulating the algorithms, or the networks, in order to generate further time savings. Bi-directional search, priority queues and contraction hierarchies are all examples of such techniques, for which Wagner and Willhalm provide a good overview in [108].

Such research has provided the background upon which the lowest cost path is found through a network for a single vehicle, in which the edge weights are fixed and known.

Whilst treating networks as static can be a useful assumption in many applications, in reality, roads are non-linear and time-varying systems and this presents problems when using the aforementioned techniques for graph search in isolation. If the user wishes to find the quickest route, then the edge weights need to reflect travel time on the edge as accurately as possible. Methods for calculating this are discussed in section 2.2.

Many of the heuristics which can be used to speed up algorithms such as A* do not take into account changes in the network over time. To address this issue Chabini and Lan present a dynamic A* algorithm in [19] for calculating shortest paths in dynamic networks. The algorithm is shown to give a five-fold improvement over an equivalent dynamic Dijkstra implementation, although the authors give no figures in comparison with the static A* algorithm. This comparison would be of interest as a static A* algorithm would be expected to complete the search much faster but give a sub-optimal solution, due to the time-varying nature of the graph being searched. Nevertheless, their work is a fundamental development for routing in dynamic networks.

In light of a better understanding of traffic dynamics and faster methods for searching graphs, researchers have attempted to find solutions to various Vehicle Routing Problems (VRPs) for some time. A first class of problems are those that can be formulated as a Travelling Salesman Problem (TSP), in which an agent (the salesman) attempts to find the shortest path between a given set of cities, visiting each destination only once. This problem has been shown to be NP-complete [80]. The same type of thinking can be applied to problems such as routing delivery vehicles in an optimum fashion. These problems come in many variants: Capacitated VRP (CVRP), VRP with Time Windows, VRP with backhauls, and VRP with pickup and delivery, all of which Toth and Vigo

(2002) provide an excellent overview of in [104].

The increasing profile of Intelligent Transportation Systems has led to new methods for increasing the efficiency of traffic travelling on roads, and vehicle route choice is one of them. Traditionally cars have only been able to make routing decisions based on static network information, but more recently cars are also able to take into account up-to-date or historical traffic data before they make their journey. The advent of autonomous vehicles also allows for stricter route control.

There is a broad range of literature covering the analytical, empirical, and algorithmic elements of vehicle route choice. In the following sections, we attempt to distil the key concepts and papers which have shaped research in this area, and led us towards formulating a novel decentralised routing approach.

2.2 Modelling Road Networks

In order to understand road networks, researchers developed models to describe the behaviour of roads at both the macroscopic and the microscopic level. Link flow models provide descriptions of macroscopic flows, that allow for road networks to be modelled at scale [77]. The development of microscopic models has enabled for some of the abstraction of macroscopic models to be discarded in favour of exact descriptions of vehicle movement and behaviour, although this comes at a greater computational cost and so has only become more popular as computers have become more powerful [4]. Here we expand on some of the research generated over the last 50 years.

2.2.1 Definition of Traffic Flow

A fundamental measure in conventional traffic analysis is the computation of the *traffic flow*. Traffic counting allows for many inferences about the state of a road network of interest, and is much more useful than taking into account either the speed or occupancy of a road individually. The time over which a traffic count is performed is an important consideration, and hence traffic counts become a traffic ***flow*** by expressing the number of vehicles per unit time. For road traffic, the common units are vehicles per hour (veh/h).

There are two main distinctions in traffic counting [99]. A *link count* is the number of vehicles passing an observation point along a road over a given period of time. The count can include traffic travelling in either one direction or both directions. A *turning movement count* is the number of vehicles making a particular turning movement at an intersection.

Considering only the case of link counting, at a generic point x_1 of a road of length X it is possible to define the traffic flow rate as the number of cars $N(x_1, T)$ that pass point x_1 over some fixed time period T. This gives us the equation for flow rate on road i (q_i) as,

$$q_i = \frac{N(x_1, T)}{T} \tag{2.1}$$

The second fundamental observation is the traffic density, which is the number of cars

per length of road. For a road of length X, with a $N(X, t_1)$ cars on it at fixed time t_1, the density of a road is,

$$\rho_i = \frac{N(X, t_1)}{X} \tag{2.2}$$

Traffic flow tells us how much a system is being used; if vehicles are able to travel at the maximum speed limit, and are packed bumper to bumper, then they are at a theoretically optimal flow rate. However, in reality, the closer vehicles are to each other, the higher the likelihood that cars will slow down and phantom jams will be created [78]. The relationship between vehicle density and flow rate is frequently studied in road dynamics.

2.2.2 Link Flow Models

A ***link flow model*** is a building block for macroscopic models. The objective of link flow models is to describe the relationship between vehicle density, traffic flow, and traffic speed. It is trivial to show that flow = density × speed, but where only flows or densities are known it is desirable to directly calculate the other values. Models take as input one of these values (usually traffic density) and return the missing values. From a routing standpoint, for a given demand and entry time to a road, the link flow model tells us the expected exit time of a vehicle. This allows for extension to describing time-varying models of roads. If we can measure the starting vehicle density on all roads, we can estimate flow between the roads over time and as such changes in the expected vehicles speed on each road over time.

This is useful, for example, if we wish to find the travel time to traverse several connected roads and have knowledge of the expected traffic density on each link. We can estimate the total travel time along each link using link flow model, and use the exit time of the previous link to model the state of the next link when a vehicle is about to enter it.

A variety of macroscopic models have been developed, such as exit functions [75], hydrodynamic models, [70], the Cell Transmission Model (CTM) [27], and bottlenecks models [107].

Link flow models are typically based on the following set of assumptions, which describe the expected behaviour of road traffic:

- ***First-In-First-Out*** (FIFO) means that a vehicle which enters a link after another vehicle cannot leave the link before it. In some real-world scenarios this is not true (i.e. overtaking), but in general, it is a fair assumption (e.g. in single lane urban networks). More importantly, FIFO also implies that *no vehicle can exit a link earlier by entering it later.* FIFO is often a required assumption in many of the models.

- ***Causality*** implies that the travel time of a user entering a link at time t solely depends on the flow of vehicles which entered the link before time t, not the vehicles after it.

- ***Conservation of Flow*** implies that the flow on a link must be the difference between cumulative inflows and outflows of that link over time.
- ***Non-negativity of Flows and Travel Times*** asserts that travel time and flow functions are positive.

These assumptions provide a framework for constructing feasible models and also asserting where models may break-down. We now present some macroscopic link flow models from the literature in more detail.

Hydrodynamic Models

The relationship between traffic density, vehicle speed, and vehicle flow has been extensively covered since Lighthill, Whitham and Richards (LWR) proposed their models relating these properties [70, 86], and authors such as Nagel and Schreckenberg demonstrated their validity [78]. Such relationships can also be shown empirically [50].

Such models describe the decrease in flow observed when the density of vehicles goes above the capacity of a road. In ***free-flow*** conditions, the flow along a road will increase linearly with the traffic density, as the number of vehicles moving at the maximum road speeds increases. Eventually, the vehicle density goes above the capacity of the road, and the velocity of all vehicles drops. In this ***congested*** state the drop in velocity reduces the flow of vehicles. This behaviour leads to a wave equation describing the propagation of a "congestion" wave along a road, against the flow of traffic. It describes many real-world phenomena, although it also predicts abrupt changes in vehicle speed which, thankfully, we do not observe in real-life. Kerner extended this model with a theory of 3-phase traffic flow, which includes the synchronization of traffic moving at slow speeds, described by Kerner as a ***wide moving jam*** [62]. We will not go into Kerner's model in detail, as it is beyond the scope of this thesis.

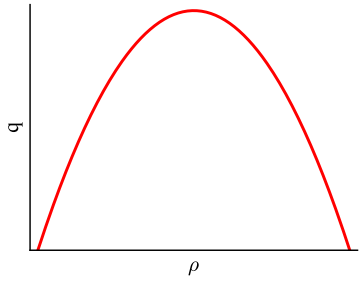


Figure 2.1: A possible relationship between q and ρ. Whilst this relationship is not realistic (noting that flow eventually becomes zero), it demonstrates the fact that flow is non-monotonic in relation to density.

LWR Model

The LWR model states that the relationship between density and flow satisfies,

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = 0, \tag{2.3}$$

where q, v and ρ are differentiable functions representing flow, traffic speed, and traffic density at time t and at the point x. Also,

$$q = f(\rho), \tag{2.4}$$

so that flow is a function of traffic density. A frequently used example is shown in Figure 2.1.

It follows that,

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial \rho} \cdot \frac{\partial \rho}{\partial x} = 0, \tag{2.5}$$

The solution to this implies that changes in the speed and density of vehicles propagate along a stream of cars as a shock wave. It describes many real-world phenomena, although it also predicts abrupt changes in vehicle speed which, thankfully, we do not observe in real-life.

Whilst the LWR model is not directly solvable, a widely used approach to applying it is the Cell Transmission Model (CTM) first suggested by Daganzo [27]. Daganzo's scheme is a discrete analogue to the LWR model of link behaviour, where freeways are divided into small cells, and difference equations are calculated at the boundaries between the cells.

Link Performance Functions

A link performance function, such as that proposed by the U.S. Beareau of Public Roads (BPR), describes travel time directly as a function of the amount of flow already on a link. The function put forward by the BPR can be useful for estimating travel times on links and therefore evaluating the effect of distributing flows on a road network. The equation is formulated such that,

$$t(x) = t_0 \left(1 + \alpha \left(\frac{x}{c}\right)^{\beta}\right) \tag{2.6}$$

where $t(x)$ is the resultant travel time for a road with free-flow travel time t_0, a constant flow capacity c, and a current flow x. α and β are parameters to be tuned to reflect the road being modelled.

Bottleneck Models

Congestion is formed in the network either on the links or at the nodes. Such a distinction can be described as the difference between ***flow congestion***, which is caused by overloading the links in the network and causing traffic to move slowly, and ***bottleneck congestion*** which is caused by a sudden drop in capacity (such as when vehicles reach an intersection) on a free-flowing road. Vickrey [107] provided the first description of such a model. In combination with the FIFO principle, it usefully describes the effect of departure time on a journey, where users must either accept delays due to being at the back of a long queue, or else leave at unfavourable times, and arrive early, to avoid delays.

Car Following Models

In contrast to the methods mentioned previously, ***car following models*** take a ***microscopic*** view of roads. These models are a form of agent-based simulation and are based on the interaction of vehicles at an individual level. A simple explanation of car following behaviour is that all vehicles maintain a certain distance from the next vehicles ahead, and will decelerate or accelerate up to the preferred maximum speed in order to maintain that distance [44,65]. Usually, this is implemented with some sort of delay to reflect driver reaction time, and additional attributes can be introduced in order to add realism to these models, such as behaviour at intersections, lane-changing, aggressiveness and braking behaviour [67]. Microsimulation is more computationally intensive than equivalent macroscopic simulation, but increases in computing power have aided in its prevalence for model validation. Various approaches can be found in the literature, although Gipp's model [44] is considered a seminal and reliable approach [61].

2.3 Traffic Assignment Theory

In order to optimise the structure of a road network designers need to understand the maximum flows that it needs to tolerate. Given the layout and demographics of an urban area, for example, the location of residential and business districts, trip distribution models provide estimates for the likely demands on the road network [84]. Road designers can use these demands to calculate the expected flows by understanding how drivers will pick their routes through the network. Traffic assignment theory studies how drivers pick their routes, based on the notion that they want to minimise some cost to themselves (usually their travel time through the network).

Traffic assignment, therefore, allows road designers to answer questions such as:

- What routes will drivers end up taking?
- What are the expected travel times?
- What are the resultant flows through the network?
- How will the network cope with congestion?

The models discussed in the previous section relate the demand on a road to its travel time. As we showed with the work of Lighthill-Witham and Richards, the capacity of the roads, in terms of maximum flow, follows a non-linear relationship with traffic density. High traffic density may result in slow speeds on the shortest route, and thus as the demand increases drivers may change their plans accordingly. This results in drivers taking alternative routes or departing at different times.

In this section, we discuss the evolution of solutions to the traffic assignment problem. There are two main distinctions between solutions, whether they are system or user optimal, and whether the approaches are static or dynamic (time-varying).

2.3.1 Static Traffic Assignment

A starting point for Static Traffic Assignment is Dijkstra's algorithm [37]. A vehicle picks the route with the shortest length, or the fastest travel time according to the maximum speed limit on each road. This is an ***all-or-nothing*** approach. Such an approach is myopic and has a high probability of certain edges being overused, with better paths available to users because, although they are longer, they are less congested.

In his seminal work, Wardrop presented two principles which have become fundamental to traffic assignment algorithms [113]. Wardrop's first principle is the case where *no driver can unilaterally reduce his/her travel costs by shifting to another route.* This defines the ***User Equilibrium (UE)*** condition, where all drivers have had the opportunity to pick the route that will best serve them, accounting also for the behaviour of other drivers. Any other route would give an equal or greater travel time. The UE condition does not necessarily lead to a ***System Optimal (SO)*** assignment. SO assignment is Wardrop's second principle that *drivers cooperate with one another in order to minimise total system travel time.* This means that, given the total demand on the network, the sum of all drivers travel times is at a global minimum.

Static User Equilibrium

The ***Static User Equilibrium (SUE)*** is the solution to the UE problem when traffic loads are constant. Sheffi [93] provides a method to find the SUE, using the following minimisation,

$$\min z(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(\omega) d\omega \tag{2.7}$$

subject to:

$$x_a = \sum_o \sum_d \sum_k \delta_{od}^{ak} f_{od}^k$$

$$\sum_k f_{od}^k = q_{od}$$

$$f_{od}^k \geq 0$$

where t_a is the average travel time for a vehicle on link a, x_a is the volume of traffic on link a, f_{od}^k is the number of vehicles on path k between the origin o and destination d, q_{od} is the trip rate between them, and δ_{od}^{ar} is given by,

$$\delta_{od}^{ar} = \begin{cases} 1 & \text{if link } a \text{ is on route k} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Static System Optimal

The ***Static System Optimal (SSO)*** solution is the SO equivalent to the SUE. Traffic flow is considered constant, and the SSO is found by solving the following minimisation (where notation is the same as for the previous minimisation),

$$\min z(\mathbf{x}) = \sum_a x_a t_a(x_a) \tag{2.9}$$

subject to,

$$\sum_k f_{od}^k = q_{od} \quad \forall\, o, d$$

$$f_{od}^k \geq 0 \quad \forall\, k, o, d$$

It is worth noting that if the effects of congestion are eliminated, the SSO and SUE solutions are equivalent and stable. In general, the effect of congestion is to ensure that the SSO is not stable when considering driver behaviour, and drivers will pick routes according to the SUE.

A Simple Example to Demonstrate the Differences Between SUE and SSO Solutions

Here it is worth providing a common example to demonstrate the SSO and SUE solutions in a small network, which is shown in Figure 2.2. This example is commonly used to demonstrate Braess' paradox [17], which is explained below.

In our network (Figure 2.2a) there is a single origin and a single destination for all traffic. Links 1-4 are 1000m long, with a flow capacity of 250 vehicles. Link 5 is 500m long with a flow capacity of 500 vehicles. We imagine a total flow of 500 vehicles between the origin and destination. There are 5 directional links in the network. Links 1 and 3 are "slow" links, with a 30 m/s speed limit, and links 2, 4 and 5 are faster links with a 60 m/s speed limit. We model the travel time of each link using the BPR function introduced

Table 2.1: Values for flow and travel time along each route in our SUE and SSO example.

	Route 1 & 2		**Route 3**		**Total Travel Time**
	Flow	Travel Time	Flow	Travel Time	
SSO	213	57	74	50.4	27998
SUE	154.2	60	191.6	60	29969
2 Routes Only	250	57.5	N/A	N/A	28747

previously so that as the flow on a link increases, and goes above that link's capacity, the travel time on that link increases.

There are 3 routes that can be followed between the origin and destination (see Figures 2.2b to 2.2d). Routes 1 and 2 are identical in length and travel time and involve taking one fast link and one slow link. Route 3 involves taking the 3 fast links.

Figure 2.2e shows the total travel time in the system, which is the objective function to be minimised when calculating the SSO, against the flow through routes 1 and 2 (as the total flow is 500 vehicles, knowing the flow in routes 1 and 2 fixes the flow through route 3). The white 'x' marks the location of the SSO solution, and the white 'o' marks the location of the SUE.

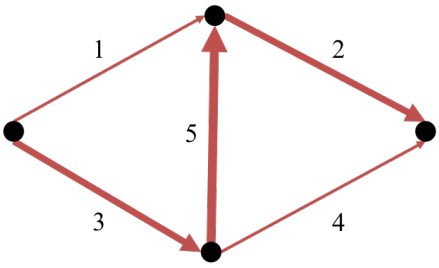
Table 2.1 details the flow along each route, the travel time for each route, and the total travel time for all vehicles. In the SUE solution, we note that the travel time on all routes has become equal, and there is no incentive for any vehicles to change its route. In contrast, in the SSO solution route 3 is significantly faster than route 1 or 2, however, both route 1 and 2 are faster than in the SUE solution.

A novelty of this example, known as Braess' Paradox is that if we remove link 5, leaving only two possible routes, then the solution is still better than the SUE when all 3 routes are available. The total travel time in this instance is indicated by the red 'o' in Figure 2.2e. This is counter-intuitive to the notion that more road capacity would mean a reduction in congestion.

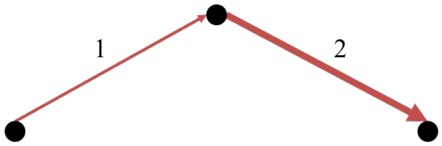
This example serves to demonstrate the differences in the SUE and SSO, and furthermore to demonstrate the possible inefficiencies in SUE solutions. Even if the SSO seems inherently unfair, the overall result can still be to the benefit of all road users.

Balancing User Equilibrium and System Optimum Solutions

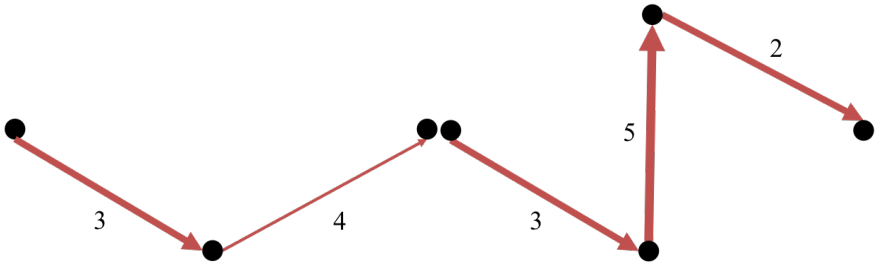
An extension of the traffic assignment problem is the balancing the user desire for taking the fastest route, with the impact that this has on the system as a whole. In taking the 'fastest route', drivers inadvertently slow down the network as a whole. Whilst the ratio of the UE to the SO can be shown to be bounded in certain instances [90], it is not known whether the change in travel time for any given driver is bounded, when comparing the UE and SO solution. Hence the SO gives no guarantees of fairness. In order to balance this Jahn et al proposed a ***Constrained System Optimal (CSO)*** problem [57], which proposed alternative routes with bounds on fairness between routes allocated to drivers.



(a) A road network with 3 routes

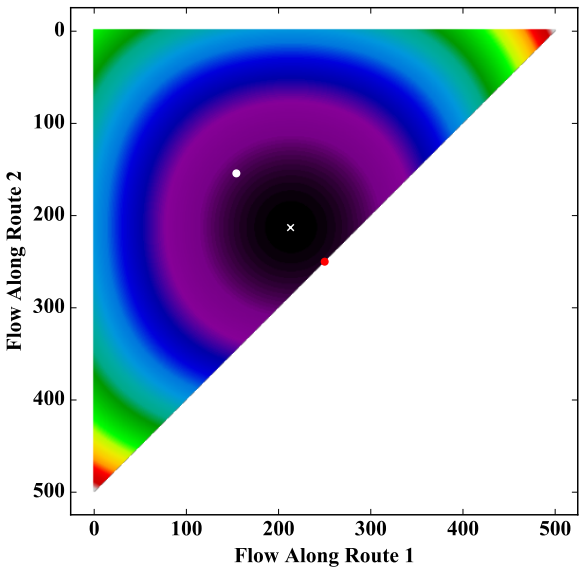


(b) Route 1



(c) Route 2

(d) Route 3



(e) Combined travel time for all drivers for a given flow split between routes 1, 2, and 3.

Figure 2.2: Example network demonstrating differences between SSO and SUE solutions. An example directional network is given, with 3 possible routes between the single origin and destination. The roads have differing speeds and capacities. A total flow of 500 vehicles must be split across the routes. The combined travel time for all vehicles, as the flow varies on routes 1 and 2, is shown on 2.2e for the SSO solution (white 'x') and SUE (white 'o'). The SSO and SUE with route 3 removed is also marked on 2.2e (red 'o'). The exact values of flow and travel time are not particularly important, only the observation that the SSO sits at a global optimum, the SUE sits somewhere off of this optimum.

The formulation for this problem is similar to that for calculating the SSO, however, the set of feasible paths is restricted to those which fall within a tolerance factor of the "normal path length", which is the path length (or travel time) the driver might have expected when taking the shortest path. Thus the CSO provides the bounds on fairness lacking in an SSO solution.

2.3.2 Dynamic Traffic Assignment

Dynamic Traffic Assignment (DTA) (or ***Dynamic User Assignment (DUA)***) extends the traffic assignment problem to dynamic networks. Specifically, road networks are considered time-varying, and the relationship between traffic density and speed leads to non-linear relationships. Whilst there is no definitive method for DTA, there are many attempts in the literature to tackle the problem in a manner which is computationally efficient, scalable, and by some measure "optimal". DTA research can be grouped into three main approaches: optimal control, variational inequality and simulation-based formulations [123].

Methods proposed in the literature for DTA can be further classified as either ***path-based*** or ***splitting-rate*** methods.

A path-based model computes a limited set of paths between each origin-destination pair, computed at some time t which would be the interval between success path calculations. Vehicles then take the calculated paths with some probability related to their calculated cost (or travel time). New paths are calculated at the next interval $t + 1$. Path-based algorithms are dependent upon an algorithm which computes a number of possible paths, for which appropriate algorithms are available and are also an area of active research [120, 121].

Splitting-rate models define turning proportions to each node in the network, specific to the destination. This splitting rate is typically defined as $f_m(d, t)$ where $f_m \in [0, 1]$ is the proportion of flow destined for node d, that uses the movement m.

Splitting rate models provide a much clearer basis for *en route* decision making, such as taking an alternative path. They also have lower memory requirements as they can describe any path on the network without storing it explicitly. However splitting rates can lead to loops, and as a splitting rate does not explicitly define a path, the travel time to the destination can only be calculated as an average from the end of the current link.

Both splitting rate models and path based models can be used to determine a UE or SO in a dynamic system. These are known as the ***Dynamic User Equilibrium (DUE)*** and ***Dynamic System Optimal (DSO)*** respectively.

Dynamic User Equilibrium

In the area of DUE, Gawron presented an early simulation and path-based approach [41]. A route is assigned to each vehicle using a shortest path calculation; the assigned routes are then inserted into a micro simulator, and the resulting traffic conditions of each edge are recorded. A portion of the vehicles are allowed to adjust their routes in order to avoid the congested sections of the network if it will improve their projected travel time.

This process is iterated so that the solution converges to a local optimum (as a rule of thumb, convergence is expected in around 50 iterations). Mahut presents a splitting-rate alternative to Gawron's path-based model [73], with a focus on finding the DUE with a reasonable error in the shortest period of time. This iterative simulation approach is also suited to genetic algorithm methods, such as those used by Sadek et al [91]. A logit model can also be used to ascribe path choice probabilistically [58].

Path-based models can have the problem that alternate routes overlap for significant portions of their length, and therefore the sections which overlap are overrepresented in the route choice probabilities. This is noted in [15], and a path-based model is presented which addresses this problem.

Dynamic System Optimum (DSO)

The DSO problem has also been addressed in the literature. The key insight to finding the DSO is that moving any vehicle to another edge should result in a total increase in travel time to all vehicles on that edge equal to the decrease in travel time to the vehicles on the edge it was previously planning to occupy. This does not negate the large computational complexity required to solve such a problem.

The assumption that all vehicles in a network are heading towards a single destination has proved useful in simplifying the DSO problem and has some applicability to considering traffic dominated by commuters to one particular area of a network. Merchent and Nemhauser [75] proposed an early discrete time macroscopic model for solving the DSO, for the single-destination problem. The formulation is non-linear and non-convex, but with certain assumptions, a global optimum can be calculated using the simplex algorithm.

Decentralised alternatives have also been proposed in the literature, with some promising results. Lim and Rus proposed a path-based algorithm for calculating DSO [71]. Taking flows on paths as the sum of probabilities that a vehicle will take a route with that road, the cost to vehicles on that route is calculated. The *marginal cost* is calculated as the total increase in travel time for all vehicles on a path for a small increase in the flow on that path. Vehicle path probabilities are varied until the marginal cost is equal for all paths. Vehicles communicate with each other to exchange information on their path probabilities, and a distributed control law is synthesised. This work was applied to 100 taxi journeys on a road network in Singapore and the results demonstrated a 15% reduction in journey times [8]. It should be noted that this work is path-based, and therefore takes a high-level macroscopic view of the road dynamics. Whilst it produces a DSO in terms of the paths, it is not necessarily a DSO in the network sense.

Optimal Control Formulations for Dynamic System Optimal Routing

A framework for the control theoretic dynamic macroscopic modelling of traffic phenomena on multi-destination road networks with time-varying (but non-elastic[1]) demands, with

[1]Elasticity in this sense indicates that demand on the network is not related to the state of the network (e.g. if the travel time for a journey doubles, this will not reduce the demand of vehicles wanting to make that journey)

application to vehicle routing, was presented by Papageorgiou [81].

The model combines a model of the nodes, which vehicles are routed between, with a model of the links, for which flow equations can be used as defined earlier. The key control variables are the splitting rates for vehicles on each route at each node. See Appendix A for further details.

A general framework for the model is to consider $\mathbf{x}$ as the state vector of the road network (we will define later what the elements of the state vector represent). The model is expressed by the general equation,

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \beta(k), \mathbf{D}(k)], \quad k = 0, ..., K-1, \tag{2.10}$$

where $\mathbf{x}(k)$ is the state vector at time step k, $\mathbf{D}(k)$ is a matrix representing the demand on the network, and β is the vector of independent splitting rates.

The performance index is expressed generally as,

$$J = \theta[\mathbf{x}(K)] + \sum_{k=0}^{K-1} \phi[\mathbf{x}(k), \beta(k), \mathbf{D}(k), k], \tag{2.11}$$

where θ and ϕ are functions to be chosen as explained below.

The performance index is to be minimised with respect to some observable metric of the system. As an example Papageorgiou considers the minimisation of travel time in the network, taking, as a heuristic, that minimising travel time means minimising a disutility function in terms of road traffic densities. In this case θ and ϕ are defined as,

$$\theta = T \sum_{m \in M} \Delta_m \rho_m(k) \tag{2.12}$$

$$\phi = T \sum_{m \in M} \Delta_m \rho_m(K), \tag{2.13}$$

where T is the sample time, and Δ_m is the length of link m, and ρ_m is the traffic density on link m. The problem can be solved by numerical methods for optimal control.

Kachroo developed Papageorgiou's work and proposed a system optimal DTA controller making use of a nonlinear H_∞ feedback control approach [60], which is known to be robust in the presence of system noise as it will minimise the maximum disturbance to the system. The control approach required simplifications to be made to the model, so that time-invariance could be assumed, due to dependence on a stationary solution to the Hamiltonian. We do not go in the minimax or Hamiltonian solutions here.

Kachroo proposed two formulations for modelling the continuous system: (i) a link-based model or (ii) a route based model [60], further details of which can be found in Appendix A. The big drawback of Kachroo's approach is that for realistic large networks there is a requirement for development of solvers for the Hamiltonian and for excellent estimation of the demand matrix ($\mathbf{D}(k)$). We are not aware of further development for this method in a realistic scenario.

More recently Ma et. al. [72] applied optimal control to the continuous-time DSO

for single-destination traffic networks with queue spillbacks. The major drawback with these formulations is the computational complexity and time required to calculate the solution, or else the required simplification of the scenarios either in terms of the model (i.e. macroscopic over microscopic) or scenario (single-origin/single-destination over multi-origin/multi-destination).

2.3.3 Sub-Optimal Approaches to Dynamic Traffic Assignment

The difficulties of calculating real-time DUE and DSO solutions have led to research in more heuristic approaches. Despite being sub-optimal, these heuristics can lead to good results, and their faster or decentralised computation makes them more realistic candidates for implementation in a real road network. We briefly summarise below some of the main approaches.

Cooperative Intersection Management is an approach whereby the individual intersections are agents who act individually or coordinate with other intersections to direct traffic in an optimal manner. The intersections may act as intermediaries for a central controller, communicate sideways (with other intersections), or only communicate with the vehicles themselves. Wang et al propose such a method of vehicle routing in [111], which they call Multiagent system based Next-Turn Rerouting (MNTR). Their main hypothesis is that current routing strategies are "reactive" and therefore unable to make decisions based on future traffic conditions. Therefore a strategy is proposed to tackle unexpected congestion events such as lane blockages and accidents. When such an event occurs a central controller signals the intersection controllers of the edge in question to begin rerouting vehicles. The intersection controller does this by calculating an optimum next turn for the next vehicle in its "queue", using a weighted cost function based on geographic closeness, travel time and traffic load. Once the vehicle has been diverted away from the affected edge it calculates a new route based on current travel times. Perronnet et. al. also use intersection controllers to solve the problem of intersection gridlock, whereby every vehicle is waiting for another vehicle to move before it can leave an intersection [83]. The method is highly centralised, as it relies on a central controller to coordinate between intersections to determine which car should be given right of way. Using the same principle as in [111], the vehicles then calculate their own routes using the A* algorithm, however, the authors perform a comparison of factors which can be used to calculate the edge weights.

A comparison of the effect of edge weights can also be found in work by Bazan et. al. on the management of driverless pod cars [11]. The proposed routing of the pod cars is done using updated travel times, however, the author's main contribution is in comparing centralised and decentralised schemes for updating edge weights, both with and without Vehicle-to-Vehicle (V2V) communication. Similar research has been done using edge weights based on both historical and stochastic factors using travel time [7] and traffic densities [115].

Kerner's work on traffic physics [63, 64] relates the probability of traffic breakdown, another term of unbounded queue growth and congestion. Kerner related traffic break-

down probability to link inflow rate and the duration of a red phase at a set of traffic lights. Based on the principle of traffic breakdown probability, Guo et. al. proposed a distributed method of minimising this probability across a road network [47, 48]. The approach increases the robustness of the network in the presence of heavy traffic loads, which in turn reduces travel times by reducing the chances of congestion in the network.

Ant Inspired Routing takes its inspiration from the way in which ants leave pheromone trails which other ants follow. The strength of these trails fades over time, hence 'fresh' routes are more attractive than 'stale' ones [34]. Ant-based optimisation has been used previously in delivery problems, but several modified approaches have been proposed in order to utilise the method for avoiding congestion. In both [23] and [29] the authors stipulate that vehicles could be routed in a manner that anticipates and mitigates traffic jams. Rather than using artificial "pheromone" trails as a way to attract vehicles, the authors agree that they can be used in order to repel vehicles away from roads which were recently used.

Whilst the authors of both papers use the ant methodology, their implementation differs in several important ways. Claes et. al. [23] use two types of virtual ant to aid in vehicle routing. The first type of ants are exploration ants, which explore possible routes in the network, attempting to detect intersections which will possibly become congested. Once a route is chosen, an intention ant communicates this route to all the intersection controllers the vehicle will pass through, so that they can update the data they provide to the exploration ants of other vehicles. The authors highlight that the intersection controllers use learning algorithms to develop the relationship between intention ants and the likelihood of congestion. Dallmeyer et. al. on the other hand use a weighted cost function for each edge, which is calculated based on the strength of the pheromone trails and the traffic density on the edge. The coefficients of the cost function must be tuned in order to attain the appropriate response from vehicles, as a poorly tuned cost function does not result in the vehicle behaviour the authors are trying to achieve. Using these updated edge weights the route is then calculated using the A* algorithm. Hasan et. al. explore route guidance based on real-time congestion information available through social media, which whilst not explicitly an ant-based algorithm, does assume disutility associated with some roads being above their optimum capacity in order to redistribute traffic [52].

An alternative to reducing travel time is to focus on the reliability of estimated travel times. Bell developed such an algorithm [13], by developing on the work of Spiess and Florian [96]. This method combines elements of path-based and splitting rate methods, building 'hyper-paths' through the network with turning rates based on how likely the driver is to encounter congestion if they turn down a particular road. These hyperpaths not only provide vehicles with alternative routes but also ensure they take the route with the highest probability of an uncongested alternative being available. This method was further developed to allow for time-varying delays and travel times [14], thanks to previous work on a time-dependent A* algorithm [19].

2.4 Validation Methodology and Performance Metrics

The majority of DUE and DSO methods use macroscopic models, with few demonstrating the use of microscopic simulation to verify the performance of the algorithm [52,60,71,81]. These models are often required in order to abstract away some of the complexities and reduce vehicles to flows between points in the network. In the case of the sub-optimal approaches, it is common to find validation via microsimulation [23,29,112]. Microsimulation is done using pre-built simulators. Simulators found in the literature include Simulation of Urban Mobility (SUMO) [111, 112] and MAINSIM [29].

In [112] the authors attempt to review the state-of-the-art in routing literature and test the algorithms in controlled conditions. The authors state that a problem with the current literature is a lack of standardised testing using comparable metrics and that this limits the scope of the reader to compare available algorithms. The authors propose testing the algorithms using real roadmaps and real traffic data.

The authors compared four algorithms, static and dynamic versions of Dijkstra's algorithm, and static and dynamic versions of the A* algorithm.

The network was taken from the TAPASCologne project [105], which exported the road layout of TAPASCologne into a format that can be run in SUMO. The project also performed DUA for an extensive set of real traffic data and has made a two hour period of vehicle routes spanning 6-8 AM publicly available. The authors broke the network down into 3 areas: (city) centre, suburban, and remote. They then sampled journeys at various scales (2, 4, 6, 8, and 10km - measured by Euclidean distance). The authors compared the algorithms both in terms of driver experience and computational efficiency.

The metrics proposed by the authors are travel time, travel distance, travel time variability, computation time, data storage, and implementation cost. Travel time variability is calculated as the ratio between the standard deviation and the average travel time for each edge over the course of the simulation, the travel time variability of a route is then the sum of the travel time variability of each edge it consists of; the purpose of the metric to measure route reliability.

The authors identify A* having a higher computational efficiency thank Dijkstra, in both static and dynamic cases, and that dynamic routing provides travel time savings over the static methods. However, Shen et. al. also state that the benefits of dynamic A* increase greatly with the trip distance. The authors suggest a threshold of 4-6 km as to where the benefits outweigh the costs of implementing dynamic methods.

Scale and location were shown to have a similar effect on the travel time variability, however, there was no obvious difference between the 4 algorithms, and so the metric was not considered successful by the authors.

Network structure may have an effect on the performance of a routing algorithm, as variation in network structure will determine properties such as the mean degree of nodes and the connectivity of the network. Braess' Paradox highlights the effects of network structure on the onset of congestion.

In the majority of cases algorithms presented in the literature are tested on contrived

grid networks [14, 29, 58, 71, 83, 111]. Only in a limited number of instances are they validated on real road networks, and a few cases where testing is done on real networks exclusively [8, 23].

In the grid networks it is not explored what the effect of properties such as the length of all the elements has on the network, however, Perronnet et. al. acknowledge that they use deliberately short links in their simulations in order to encourage congestion in the links.

Traffic load and trip distribution could also be an influencing factor in terms of routing algorithm performance. In cases of real road networks where data is available, authors such as Wang [110] and Lim [71] have used actually recorded traffic demand for highly realistic traffic loads.

Where data is not available or the network has been contrived, trips are generated according to some model. Perronnet et. al. use additional nodes connected to the edges of the network to act as car origin and destinations, they then generated cars by varying between a low and high car generation rate[2], and varying between homogeneous and non-homogeneous trip distribution. In contrast Wang et. al. state that they generate traffic demand uniformly on their synthetic networks, but fail to show if exactly what they mean by this (e.g. a constant car generation rate, or a homogeneous trip distribution).

Dallmeyer et. al. use random Origin-Destination (O-D) pairs in their grid network, an approach which would encourage non-homogeneous traffic loading. Once the network has 'settled', they then use the novel strategy of holding the number of vehicles in the network at a constant level, only generating a new vehicle when one reaches its destination. This approach is inherently unrealistic, but the authors vary the number of vehicles in the network, and so are able to demonstrate the performance under different traffic loads.

Travel time is the most frequently used metric in the literature. This is usually measured as a mean or total for all journeys [7, 8, 11, 23, 29, 71, 111, 115], although in [112] the authors do a good job of only comparing journeys of a similar distance. Wang et. al. also use **travel distance** as an indicator of financial cost to the user, equating distance travelled with fuel used.

In contrast, Perronnet et. al. only use the **number of vehicles in the network** to compare between the routing strategies tested in their paper.

2.5 Discussion

Research into cooperative traffic routing has had several major inputs in the last few years. Many of the authors have demonstrated promising methodologies and demonstrated a sophisticated understanding of both the dynamics of road networks and the computational hurdles that need to be overcome. In particular, those papers which encourage the prediction and mitigation of congestion [23, 29, 111] are extremely promising.

[2]Car generation rate refers to the number of cars entering the network at each time step

However, it is almost impossible to compare the relative performance of these algorithms from the results reported in each individual paper as different metrics are typically used for their validation. Within the literature there is a lack of consensus about what performance means; travel time, travel distance, system stability and environmental cost are all cited as factors which can be improved upon over the current situation, but many authors fail to provide a definitive value on their work. A definitive set of network tests, traffic loads and metrics would greatly improve the field. This would also enable researchers to better classify their work under their aims, such as savings in the environment or the economy.

Chapter 3

Intersection Control: An Overview

In this chapter, we review the literature on the subject of intersection control. Firstly we establish terminology for describing the topology of an intersection. We then cover intersection control strategies and distinguish between the main classifications of strategy. We contrast the state-of-the-art in centralises traffic responsive methods which promising decentralised approaches, which may offer the benefits of reducing congestion, but in a manner which makes them much more scalable and robust.

Traffic light controlled intersections are a fundamental aspect of modern road networks. The control system is responsible for managing the crossing or merging of conflicting traffic streams, and in urban areas, the performance of an intersection has a measurable effect on congestion in the network as a whole. Electric traffic signals can be traced back to 1914 [42]. As explained in The *Traffic Control Systems Handbook* [45], since 1952 cities have increasingly relied on computer-aided signal control. The introduction of detectors into road infrastructure enabled both the actuation of traffic lights and also the collection of data to develop better timings.

The performance of traffic lights at a network level has always been limited by available computing power and the models being used. As both of these have improved over time, increasingly sophisticated methods have been developed.

3.1 Terminology Describing an Intersection

The development of signalised intersections has led to some generally accepted formal terminology, however, there are some differences between sources. Here we clarify some terminology we will be using, which can be found in other literature on the subject [106]. We imagine any intersection as having a number of ***input links*** (roads for vehicles to enter the intersection) and a number of ***output links*** (roads for vehicles to leave the intersection). A ***stop line*** marks the end of an input link. The maximum flow rate of vehicles crossing the stop line of an input link is termed the ***saturation rate***.

A ***phase*** is a movement from an input link to an output link, made by crossing the stop line. A ***queue*** is a queue of vehicles; the vehicles may all belong to the same phase or may belong a mix of phases with the same input link. A set of simultaneously permitted

phases is termed a ***stage***. This topology is visualised in Figure 3.1.

The ***cycle-time*** is the time it takes for the intersection to go through every stage in a fixed-cycle program. The ***green time*** is the length of time that the light is green for a phase, and the ***red time*** is the length of time that the light is red for a phase. ***Loss*** is time spent with no vehicles moving, primarily due to amber stages of the traffic lights. The ***green split*** is the proportion of green time allocated to each stage in a cycle.

When networks of intersections are considered we may also consider the ***offset*** between traffic lights, which is the delay between green lights for a downstream and upstream link. We consider a network to be ***under-saturated*** if all the queues in the network only grow during a red light. When queues grow during their green phase then we can say that the network is ***over-saturated***.

Clever design of the offsets allows for vehicles to experience a ***green-wave*** in under-saturated traffic conditions, which is a continuous set of green lights across adjacent intersections. In saturated conditions, the most efficient approach is ***store-and-forward***, where queues in downstream links are given a green light before the upstream queues, in order to create space for new vehicles.

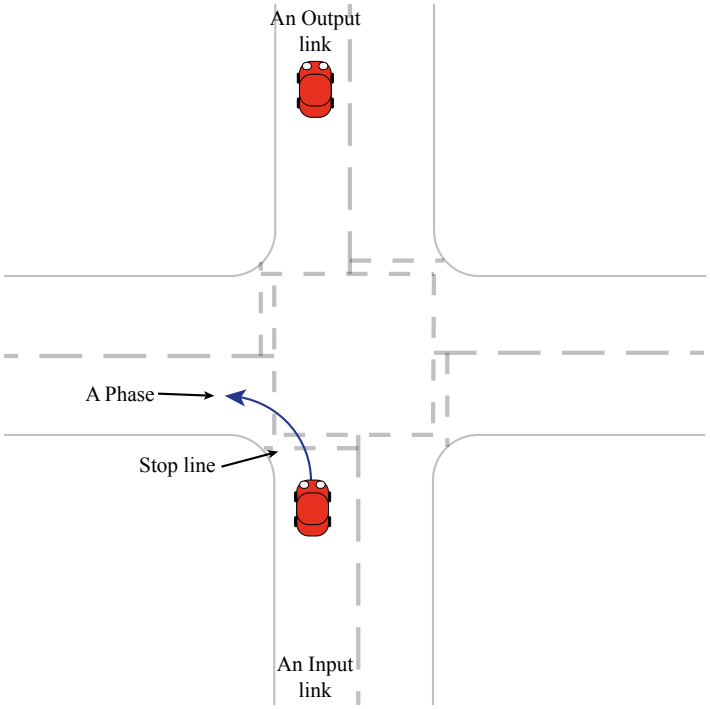
3.2 Intersection Control

Control theory, in the realm of intersection control, relates to selecting the stage and the stage duration at any particular time step, in order to minimise or maximise some objective function [55]. There is no clear consensus on the objective function that should be selected, delay at an intersection is an obvious candidate, but it could also be to maximise the flow along a road, reduce the travel time through a network, or minimise vehicle emissions. In a review of the roles of traffic detection, optimisation objective and control feedback it was found that the key state variables are queue lengths and mean arrival rates, rather than delay [54]. These findings are backed up by the performance of control schemes which work well in saturated networks [2, 3, 32]. The use of delay equations to optimise traffic seems to have stagnated somewhat, with limited work having been done to develop the accuracy and range of application of queuing models, and develop new models [100].

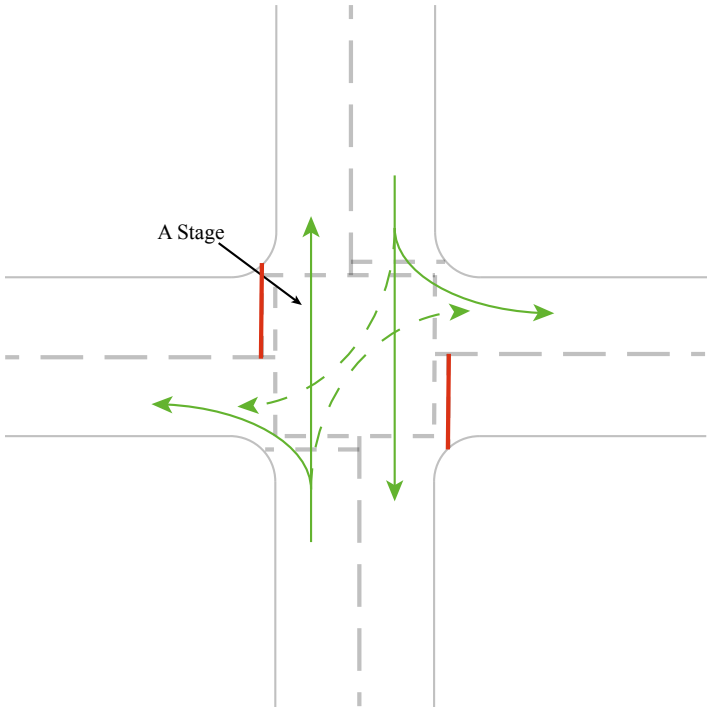
3.2.1 Existing Approaches to Signal Timings at Intersections

The solution space can be divided into approaches which are either ***fixed-time*** (or ***fixed-cycle***) or ***traffic-responsive***, and are either ***isolated*** or ***coordinated***. Furthermore a solution can be ***stage-based*** (stages have a pre-determined order) or ***phase-based*** (stages can be implemented in any order) [82]. We briefly describe the key features of these approaches.

The simplest approach to traffic lights is the fixed-cycle. Stages are activated for a fixed period of time in predetermined order. A fixed-cycle scheme can take different approaches to try and improve performance. For example, in under-saturated conditions, the neighbouring intersections are offset so that the green light of the next intersection begins once the vehicles from the previous intersection have travelled the distance between



(a) A phase is a movement from an input link to an output link



(b) A stage is a combination of compatible phases

Figure 3.1: A typical 4-way intersection with 4 input-links, 4 output-links

them. In a saturated network, a negative offset is used, so that the green light at the next intersection is given early in order to clear traffic and make space for the next set of cars, thus avoiding the propagation of congestion when vehicles have a green light but nowhere to go.

Fixed-cycles must be chosen in advance, using predictions of demand and some model of traffic behaviour. Their design can be based on centralised or decentralised optimisation, but it will always be done offline. This is likely to lead to be a suboptimal solution [6].

A traffic-responsive strategy is calculated online [88]. It may be similar to the fixed-cycle, in that some timings have been calculated in advance, however real-time measurements of the traffic state are taken and adjustments made to the green split, cycle length, and/or offsets. The goal is optimisation in real-time, and the optimisation function will depend on the strategy being implemented.

If the strategy is isolated, then it attempts to optimise the signal timings for a single intersection [5]. In contrast, a coordinated strategy groups intersections together into a single model [89], and must find an optimal solution for the network as a whole.

If the strategy is stage-based then it determines the optimal green split and cycle-time for a pre-specified program, however, if the strategy is phase-based then it additionally determines the optimal order of the stages [55].

3.2.2 Isolated Fixed-Cycle Approaches

Isolated fixed-cycle strategies are those which aim at controlling a single intersection using predetermined signal settings. An early optimisation objective was to minimise vehicle delays at an intersection. Webster's formula [69, 114] provided a method to estimate the average delay per vehicle at an intersection, based on the cycle time, green time, arrival rate, saturation rate and capacity of the intersection. It can be given as,

$$d = \frac{c(1-\frac{g}{c})^2}{2(1-x\frac{g}{c})} + \frac{x^2}{2q(1-x)} - 0.65(\frac{c}{q^2})^{\frac{1}{3}}x^{(2+5\frac{g}{c})} \tag{3.1}$$

where d is the average delay per vehicle, c is the cycle time, g is the green time within each cycle, r is the red time within each cycle, x is the degree of saturation (ratio of flow to capacity), and q is the arrival rate of vehicles.

The three terms of the right-hand side of equation (3.1) model delay to uniform arrivals, delay due to random arrivals, and an empirical adjustment, respectively. Webster's formula is well suited to manipulation when considering, for example, uniform arrival rates only. The formula itself provides a possible objective function by which engineers can optimise fixed-cycle control schemes. A fixed-cycle strategy, SIGSET [6] does just this and minimises the total intersection delay for a given number of demands at an intersection. In contrast, another approach known as SIGCAP [5] maximises the intersection capacity, which can be done by optimising to maximise the demand, rather than minimising the delay. Such an approach will always yield the highest allowable cycle time. SIGSET and SIGCAP are examples of stage-based solutions, in that they took a predetermined set of ordered stages and determined the green time which should be allocated to each stage,

however, the problem has also been extended to consider the more complex phase-based solution which determines the order of the stages themselves [55].

3.2.3 Coordinated Fixed-Cycle Approaches

Coordinated fixed-cycle approaches extend methods used for solving the isolate fixed-cycle problem to groups of intersections.

TRANSYT [89] was developed to optimise a network of coordinated fixed-time intersections. Using offline traffic data, knowledge of the network, and initial signal settings, TRANSYT optimises for a Performance Index (PI) based on delays and stops. The cycle time is fixed (and chosen by the user), but offsets and green splits are adjusted by iteratively optimising them based on some starting value.

These fixed-cycle schemes have an inherent weakness in their inability to adapt to changing traffic demands. Incidents which disrupt traffic flow, short-term variations within a day, and long-term changes in demand may cause such models to break down.

3.2.4 Coordinated Traffic-Responsive Approaches

Coordinated traffic-responsive strategies address some of the limitations of strategies such as TRANSYT, by adapting in real-time to measured traffic data. Split Cycle Offset Optimisation Technique (SCOOT) [88] is the traffic-responsive extension of TRANSYT. It is used in 150 cities in the United Kingdom [82]. SCOOT determines the effects of changing the offset, splits or cycle time at individual intersections, and pushes those changes to local intersections if it determines them to be beneficial. SCOOTS performance is known to deteriorate in saturated traffic conditions, although features to combat this can be found in [45].

A similar set-up is found in the Sydney Co-Ordinated Traffic System (SCATS) [94]. SCATS differs from SCOOT in having both a strategic and a tactical control layer. Strategic control is regional and links together intersections, which share a common cycle time, and sets the splits and offsets. The tactical layer, which is based at the intersection, may adjust a stage so that certain phases are omitted, terminated early or extended. In both SCOOT and SCATS the cycle time is adjusted in order to keep the degree of saturation of the lane with the highest degree of saturation below 90%. The green splits for both methods are set to reduce delays, but SCATS equates this to balancing the saturation at approaching lanes to an intersection, whereas SCOOT computes this explicitly.

Where SCOOT and SCATS *react* to measured traffic data to reduce some PI, other methods *model* traffic in order to predict trends, and optimise for both current and future demands. Such methods disregard splits, offsets and cycles, and instead rigorously calculate switching times for each intersection, based on the performance of the network over some fixed time horizon (e.g. 60 seconds). Rolling horizon approaches are similar to model predictive control, but only optimise up to some fixed (rolling) time window. A dynamic programming approach is taken, and the algorithm works back from the end

of the time horizon to calculate the settings for the next step. The resulting settings are applied for some much short time period (usually a few seconds) and then recalculated.

Models which use a rolling horizon methods include RHODES [76], OPAC [40], and PRODYN [53]. The most recent of these, RHODES, is a prediction and control architecture which works at both the network and the intersection level. At a network level, the model captures the slowly changing loads on the network, taking into account factors such as route choice and road closures. Predicted loading on each link allows RHODES to allocate green time for each phase in the network. Given the approximate required green times, RHODES adjusts the change of the stage at the intersection level based on observed and predicted vehicle arrivals. In contrast to centralised methods to control intersections as a network, these methods split the problem into sub-networks, and so demonstrate a move towards more decentralised decision making.

Traffic-responsive approaches can also be applied to bandwidth problems in arterial roads [30]. In this methodology, bandwidth is controlled using signal offsets and variable speed limits. Furthermore, the myopic aim of only maximising bandwidth is addressed, and the scope of the optimisation problem is increased to include energy consumption and network travel time. It is proposed that maximum speed does not correlate with minimum travel time, but does correlate with maximum energy usage. A solution is formulated to the two-way maximum bandwidth problem. The methodology is validated via numerical methods, to show increased theoretical bandwidth, and simulation in AIMSUN, to show microscopic vehicle behaviour. The key results indicate that reductions in energy consumption (due to reduced stops and speed) can be made without a reduction in travel time.

3.3 Decentralised Traffic-Responsive Approaches with Coordinated Behaviour

Clearly the most likely models to find a network optimal solution for a given cost function are traffic-responsive coordinated phase-based strategies, however, these models are frequently complex, with millions of permutations in complex networks that cannot be solved in reasonable time-frame when considering the real-time updating of the traffic lights. The opportunity offered by costly but powerful computation and distributed sensing means that we find a wide range of strategies both in the literature and used in practice. Here we provide a chronological development of the best strategies in the literature. We then branch into the state-of-the-art, showing the range of approaches which have touched on optimal control, game theory, machine learning and communications science.

There are many examples of attempts to further decentralise intersection control using various methodologies. One approach is to try and mimic the performance of a human controller at an intersection, which can be achieved using q-learning [16]. This approach yields good results but its performance was only considered for an isolated intersection. In contrast in [1] a q-learning algorithm is applied in several groups (termed holons) of intersections of increasing size, in an attempt to distribute decision making with a coherent

overall strategy. Other decentralised approaches include utilising game-theory [10] and Bid-Based Control [56]. Bid-based control is a scheme in which drivers may bid for priority at an intersection, where a movement manager manages the overall bid for a given queue of drivers. Regardless of the moral implications of a system which favours those most able to pay, this economic driven approach was able to balance delays in comparison to actuated traffic lights at a single crossroad intersection where one approach had much higher traffic flow.

The future of intersection control may do away with traffic lights altogether [35, 36]. An example is slot-based intersection control, where driverless vehicles with ***Vehicle-to-Infrastructure (V2I)*** communication request a time slot to pass through the intersection, and adjust their speed accordingly to pass through at the correct time. As the vehicle localisation and control improves it will be possible for vehicles to pass through from opposing directions simultaneously, whilst avoiding any collisions. In [97] such a system was modelled at a city-scale, including pedestrian and stages for human drivers, and found that traffic capacity might be doubled were such a system implemented. The problem of how conflicting reservations may be resolved whilst minimising the risk of vehicle collisions has been the subject of much recent research and indicates that this is a technology which is gaining moment [21, 28, 38, 49, 68]. Unfortunately, such systems are unlikely to be implemented at a large-scale soon. The introduction of the first driverless vehicles to our roads may not be far off, but based on the introduction of previous technologies it will take a long time for them to saturate the market. In the meantime, there are good reasons to introduce technologies that can take advantage of V2I communication or improvements in image recognition and vehicle sensing.

3.3.1 The Principle of Work Conservation: A Promising Decentralised Method for Intersection Control

Work conservation, for a traffic light controlled intersection, means that for a system with some zero and some non-zero queues, a non-zero queue will always be actuated. Work conserving controllers naturally infer that less of the cycle time is wasted servicing empty queues. Work conserving traffic lights are also desirable because they are completely decentralised, meaning they do not require a centralised controller and no coordinating strategy. Where a centralised policy must be changed when a new intersection is added to the network, a decentralised traffic light adapts automatically to the change. The system can, therefore, be implemented in stages, is scalable, and is adaptable at the level of the individual intersection.

The properties of work-conserving controllers have been investigated thoroughly in the literature and using network calculus it has been shown that for any network of intersections for which a fixed-interval control scheme can keep queue lengths and delays bounded, a work conserving controller exists that also keeps queue lengths and delays bounded [20, 25, 106]. One such algorithm which resulted from this work was the max-pressure controller, which uses the queue lengths as weights to determine the service policy at the intersection. Extensions proposed to this are analogous to the back-pressure routing

algorithm discussed for communication networks in [39, 43, 98]. Whilst the algorithm does not require vehicle arrival rates, it is assumed that ***turning ratios***, which provides the expected number of vehicles along each movement, can be accurately estimated.

The benefits of the backpressure algorithm have been further verified in the literature for road networks. For example, algorithms have been designed which enable maximum network throughput, and results were verified in MITSIMlab, where it was able to outperform SCATS [116]. In the case where stages are strictly ordered and turning rates could not be computed exactly further backpressure policies have been shown to still retain the stability properties [66]. This work was extended to the case where queue measurements, saturation rates, and turning ratios are all estimated or noisy [119], and still, it was possible to show that maximum throughput is achieved under certain conditions. In this case, the algorithm was verified in the microsimulator VISSUM, showing improvement over a fixed-cycle strategy based on the optimisation of Websters formula [118].

A criticism of the above methods is that they do not account for downstream congestion, which causes the stability proofs of work-conservation to break down [46]. A capacity aware back-pressure controller was proposed [46] as a counter to those examples where supposedly work conserving controllers, such as those in [66, 106, 119], can lose work conservation in the presence of gridlock, when the input links do not have equal maximum queue lengths at an intersection. The algorithm proposed was a modified back-pressure algorithm, where pressures on all roads are 'normalised' using a convex function, rather than using absolute queue lengths and saturation rates to determine weights. The performance of the algorithm was verified via simulation (SUMO) [46].

The benefits of backpressure controlled intersections have been proven in the literature both mathematically and via simulation, however, there are many possible permutations of the algorithm. In particular, there has been much research into its performance when turning ratios cannot be measured exactly, but we have not found anything in the literature about the benefits of determining the exact numbers of vehicles and allocating pressure precisely. Green time is also either held constant [46] between subsequent calculations of the backpressure or else is based on the backpressure weighting of the stage in question [66, 117]. There is scope for research into a combination of adaptive green times using traditional control theoretic approaches, in combination with stage selection based on backpressure or max-pressure.

3.4 Discussion

Research into intersection control has evolved steadily over the last 40 years. The offline optimisation of a single intersection [5] has been surpassed by the traffic-responsive online optimisation of groups of intersections [88]. However, there are drawbacks in terms of computation overhead, ability to scale, and robustness to change that state-of-the-art centralised strategies present.

The principle of work conservation has been shown to be extremely effective when designing decentralised control strategies, which avoid some of the aforementioned problems. However, when work conservation breaks down this can cause network congestion,

and the breakdown is clearly dependent on network topology [46]. Further research into work-conserving intersection controllers would, therefore, be of benefit.

Chapter 4

A Decentralised Routing Algorithm for Enhancing Network Resilience to Congestion

It is postulated that autonomous vehicles will decrease private car ownership, instead encouraging users to utilise cheap fleets of autonomous vehicles [101]. In a healthy market, it is probable, and desirable, that multiple operators will compete for market share. Such operators will be unlikely to share proprietary routing algorithms, and it may be difficult to coordinate centralised routing strategies, or even decentralised strategies if they are overcomplicated. Current trends in the industry suggest that sub-optimal User Optimal Dynamic Traffic Assignment will be adopted as a feasible solution. In this chapter, we present a simple decentralised routing algorithm, which requires only one-way communication with infrastructure, and will be shown to enhance network resilience to congestion in a number of circumstances.

In our approach a vehicle chooses the next road at every intersection by minimising a cost function that combines both distance to the destination and congestion of each road that a vehicle could take at its next turn. In contrast to methods discussed in Chapter 2, our method only requires short-range communication between vehicles and an intersection controller (see Fig. 4.1).

The initial inspiration for the decentralised controller was the method of coverage control, in non-linear control where decentralised agents distribute themselves optimally over an area to be measured [92]. Using inspiration from centralised control formulations, which optimise traffic flows based on a disutility function of the traffic density, we propose to have vehicles self-distribute in a manner which evenly spreads their density over a road network, whilst still taking them towards their destination.

The algorithm we present is unique because it combines Dynamic Traffic Assignment (DTA) in a decentralised and localised manner with no Vehicle-to-Vehicle (V2V) communication and one-way Vehicle-to-Infrastructure (V2I) communication.

- Our algorithm does not increase the average travel time of vehicles in uncongested networks, therefore it poses no disadvantage to users in this case.

- Our algorithm increases the network resilience to congestion.

The key properties of our algorithm are that it:

- Is a decentralised routing algorithm which attempts to perform System Optimal Traffic Assignment. This is in contrast to algorithms which are primarily centralised mixed-integer dynamic programming assignments.
- Is a dynamic routing algorithm that is not affected by the road network being modelled as a time-invariant non-linear system.
- Requires local occupancy data of the next available road.
- Requires one-way communication. Vehicles only receive data and make a routing decision based on our algorithm, they do not need to transmit back.
- Has a low overhead for computation resources
- Has $\mathcal{O}(1)$ complexity for an increasing number of vehicles in the network

4.1 Problem Formulation

We consider the problem of routing a set of N vehicles, $\mathbf{I} = \{1, 2, ..., N\}$, travelling in a bounded region $\mathbf{Q} \subset \mathbb{R}^2$. The starting position of the i-th vehicle is denoted $\mathbf{s}_i \in \mathbf{Q}$, $i \in \mathbf{I}$. The position of the i-th vehicle at time t is denoted by the point $\mathbf{p_i}(t) \in \mathbf{Q}$, $i \in \mathbf{I}$. The intended final destination of the i-th vehicle is denoted by some point $\mathbf{d_i} \in \mathbf{Q}$, $i \in \mathbf{I}$.

Vehicles are located on a road network $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ contained in $\mathbf{Q}$, where $\mathbf{V} = \{1, 2, ..., m\}$ is the set of m vertices which represent junctions, and $\mathbf{E}$ is the set of edges which represent the roads between junctions. The j-th junction is labelled v_j, $j \in \mathbf{V}$. An arbitrary road between two junctions v_j and v_k is denoted $(v_j, v_k) \in \mathbf{E}$, where v_j and v_k are the parent and child vertices (junctions) of the edge (road) respectively.

We propose that, in order to decide what route to take, the i-th vehicle approaching junction v_j calculates a cost function $J_i^{(j,k)}$ for each road $(v_j, v_k) \in \mathbf{E_j}$, where $\mathbf{E_j}$ is a set of all roads departing from the junction v_j. We assume the cost function for the road is the weighted sum of several terms represented in a vector $\mathbf{\Phi}_i^{(j,k)}$, which will be defined later in Sec. 4.2, and denote $\underline{\boldsymbol{\alpha}}$ the vector of weights, such that,

$$J_i^{(j,k)} = \underline{\boldsymbol{\alpha}}^T \cdot \mathbf{\Phi}_i^{(j,k)} \tag{4.1}$$

The vehicles choice of road at junction v_j is then the one that minimizes the cost function, i.e. it is such that,

$$\min_{(v_j,v_k)\in\mathbf{E}_j} J_i^{(j,k)} \tag{4.2}$$

Then the crucial problem is to design cost functions and tune their parameters in order to achieve the desired global behaviour in the road network. For example, the desired

behaviour could be to reduce the mean travel time for all vehicles, reduce the standard deviation in delay between vehicle journeys (i.e. homogenise travel times) and/or other objectives related to vehicle journeys.

In this chapter, we are most interested in characterising the network resilience to congestion, i.e. the maximum rate at which vehicles may enter the network before the majority of vehicles experience a significant increase in delay to their journey.

Before presenting our decentralised solution to the routing problem, we first give some definitions and introduce notation that will be used throughout the rest of the Chapter.

Let $\mathcal{D}(v_j, v_l)$ represent the shortest travel time between two vertices in the network, calculated using Dijkstra's algorithm [37] with free-flow travel time as the edge weights. The minimum expected travel time for the i-th vehicle, is therefore $\mathcal{D}_i = \mathcal{D}(\mathbf{s}_i, \mathbf{d}_i)$. The estimated travel time along the shortest path does not include delays due to traffic lights or turning movements. The actual measured travel time for a vehicle on its journey is termed $\mathcal{T}_i = \mathcal{T}(\mathbf{s}_i, \mathbf{d}_i)$. The actual travel time is recorded from the moment the vehicle wishes to enter the network, even if it is not able to do so until a later time step, in this manner the travel time also incorporates any 'depart' delays. The delay experienced by the vehicle along its route is termed ω_i and is calculated as,

$$\omega_i = \mathcal{T}_i - \mathcal{D}_i \tag{4.3}$$

The mean expected travel time for all vehicles is,

$$\bar{\mathcal{D}} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathcal{D}_i \tag{4.4}$$

and the mean travel time measured for all vehicles,

$$\bar{\mathcal{T}} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathcal{T}_i \tag{4.5}$$

implying that the mean delay experienced by all vehicles is,

$$\bar{\omega} = \bar{\mathcal{T}} - \bar{\mathcal{D}} \tag{4.6}$$

We denote the number of cars entering the network at every time step as λ, and consider that the mean delay is a function of this, i.e. $\bar{\omega} = \bar{\omega}(\lambda)$. We further define the delay as acceptable if it is less than some delay threshold, termed $\hat{\omega}$, where,

$$\hat{\omega} = \beta \cdot \bar{\mathcal{D}}, \quad \beta \in \mathbb{R}^+, \quad \beta \geq 0 \tag{4.7}$$

and note that $\bar{\omega} \leq \hat{\omega}$ implies that,

$$\bar{\mathcal{T}} \leq (1 + \beta)\bar{\mathcal{D}}, \quad \beta \geq 0 \tag{4.8}$$

where β is the acceptable ratio between the minimum expected travel time for a vehicles

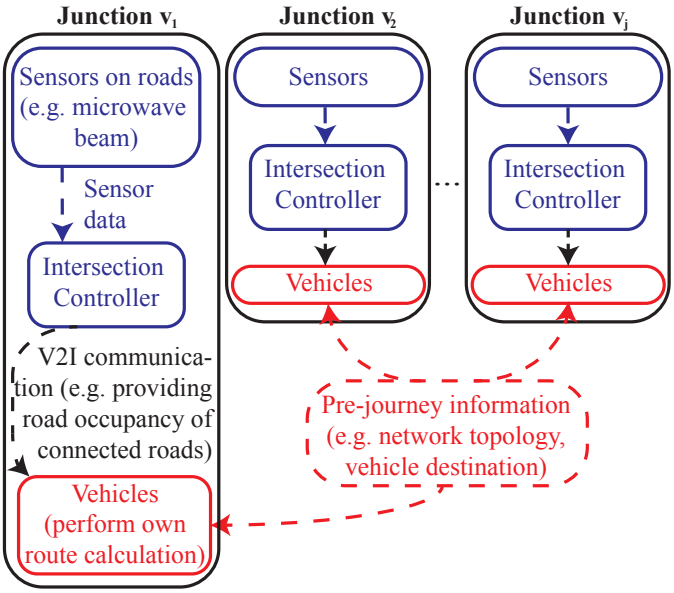


Figure 4.1: The communications hierarchy required to implement our routing algorithm

journey, and the actual delay it experienced.

The maximum car generation rate before vehicles experience intolerable delays is termed $\hat{\lambda}$, and is defined as,

$$\hat{\lambda} = \max\{\lambda \ : \bar{\omega}(\lambda) \leq \hat{\omega}\} \tag{4.9}$$

4.2 The Decentralised Routing Algorithm (DRA)

To solve the problem above we propose to use an intersection controller at each junction to keep track of the following properties for each road (v_j, v_k) connected to it:

1. A load $[L^{(j,k)}(t)]$ - modelling the number of vehicles currently using the road (v_j, v_k).
2. A capacity $[C^{(j,k)}]$ - representing the maximum number of vehicles that can fit onto road (v_j, v_k).
3. An occupancy $[\eta^{(j,k)}(t) = \frac{L^{(j,k)}(t)}{C^{(j,k)}}]$ - modelling the percentage of space on road (v_j, v_k) occupied by vehicles.

When a car, say the i-th vehicle, reaches a junction v_j, it can access two types of sensory functions $\phi_i^{(j,k)}$ and $\rho^{(j,k)}$, for every road $(v_j, v_k) \in \mathbf{E_j}$. The first sensory function, $\phi_i^{(j,k)}$, is related to the minimum estimated travel-time for the car i to reach its destination, d_i, if it takes a particular road $(v_j, v_k) \in \mathbf{E_j}$:

$$\phi_i^{(j,k)} = \phi(d_i, v_j, v_k), \quad 0 \leq \phi_i^{(j,k)} \leq 1 \tag{4.10}$$

The second sensory function, $\rho^{(j,k)}$, is related to the occupancy $\eta^{(j,k)}(t)$ of road $(v_j, v_k) \in \mathbf{E_j}$

at time t:

$$\rho^{(j,k)} = \rho(\eta^{(j,k)}(t)), \quad 0 \leq \rho^{(j,k)} \leq 1 \tag{4.11}$$

To obtain $J_i^{(j,k)}$, expressions (4.10) and (4.11) are combined via a control parameter $\alpha \in (0, 1]$, so that when car i reaches junction v_j, the cost functions for every road $(v_j, v_k) \in \mathbf{E_j}$ can be computed as,

$$J_i^{(j,k)} = \begin{bmatrix} \alpha & (1-\alpha) \end{bmatrix} \begin{bmatrix} \phi_i^{(j,k)} \\ \rho^{(j,k)} \end{bmatrix} \tag{4.12}$$

Note that by tuning the control parameter α we can make the vehicle routing choice more or less sensitive to distance or congestion respectively.

4.2.1 Considering the Variation of α and its Effect on Routing

Here we study analytically how the value of α will cause a vehicle to change its route. Consider the case of two possible paths 1 and 2, whose costs are the sum of the generic monotonic increasing functions ρ_1, ϕ_1, ρ_2 and ϕ_2 respectively. Let us assume that the cost of path 1 is lower than path 2, and as such will be taken by the vehicle. We do not assume any other characteristics of the network. We describe this using the following inequality,

$$\alpha \cdot \rho_1 + (1-\alpha) \cdot \phi_1 < \alpha \cdot \rho_2 + (1-\alpha) \cdot \phi_2, \quad \alpha \in (0, 1) \tag{4.13}$$

Lemma 1. *When two paths are of equal length, the one with the lowest occupancy will be taken.*

Proof. It is trivial to show that when $\rho_1 = \rho_2$ equation (4.13) reduces to $\phi_1 < \phi_2$. Similarly if the cost of route 2 was less that route 1 we would find that $\phi_1 > \phi_2$. □

Lemma 2. *When two paths are of equal occupancy cost, the one with the shortest path will be taken.*

Proof. As before it is trivial to show that when $\phi_1 = \phi_2$ we must have $\rho_1 < \rho_2$. □

We can rearrange the inequality (4.13) to show that,

$$\alpha < \frac{C}{C+1}, \tag{4.14}$$

where,

$$C = \frac{\rho_2 - \rho_1}{\phi_1 - \phi_2} \tag{4.15}$$

We apply conditions so that we are considering the case where the route with the lower overall cost, 1, has a higher route cost, but a lower occupancy cost,

$$\phi_1 > \phi_2$$
$$\rho_1 < \rho_2$$

s.t. $C > 0$.

Lemma 3. *As the difference in route cost between two routes approaches* 0*, the value of* α *required to detour onto the one with the lower occupancy cost will approach* 1 *from below.*

Proof. We consider the asymptotic behaviour of (4.15). As $\phi_2 \to \phi_1$, then $C \to \infty$, and hence from 4.14 $\alpha \to 1$. □

Lemma 4. *As the difference in occupancy cost between the two routes approaches* 0*, the value of* α *required to detour onto the one with the lower occupancy cost will approach* 0 *from above.*

Proof. As before, we consider asymptotic behaviour. As $\rho_1 \to \rho_2$ we find that $C \to 0$ and from equation (4.14) $\alpha \to 0$. □

We can conclude that the longer the detour we wish it to be possible for a vehicle to take in the presence of equal occupancy differences between two roads, the lower the value of the tuning parameter α must be. Whilst this conclusion is obvious, in our system where we normalise the cost functions we can visualise the value of α that will allow for a detour. In Section 4.3 we show numerically that α can be tuned to modify the performance of the Decentralised Routing Algorithm. We then compare this with a method for estimating the value of α which will perform best.

4.2.2 Choice of Sensory Functions

We propose to choose the sensory function $\phi_i^{(j,k)}$ as follows

$$\phi_i^{(j,k)} = \frac{\mathcal{D}(v_j, v_k) + \mathcal{D}(v_k, d_i)}{\max_{v_s, v_t \in \mathbf{V}} \mathcal{D}(v_s, v_t) + \max_{(s,t) \in \mathbf{E}} \mathcal{D}(v_s, v_t)} \tag{4.16}$$

In this cost function the numerator is the sum of the travel-time of the road being considered, $\mathcal{D}(v_j, v_k)$, and the travel-time of the shortest path (found using Dijkstra) between junction v_k and the destination d_i, $\mathcal{D}(v_k, d_i)$), and hence is equivalent to the shortest path from the i-th vehicles current position, v_j, to its destination, d_i, via the junction v_k. The denominator is the sum of the network diameter (or the travel time between the two most distant nodes in the network), $\max_{v_s, v_t \in \mathbf{V}} \mathcal{D}(v_s, v_t)$, and the longest edge in the network $\max_{(s,t) \in \mathbf{E}} \mathcal{D}(v_s, v_t)$, which ensures $0 < \phi_i^{(j,k)} < 1$.

To allow for the effect of congestion[1], the sensory function $\rho^{(j,k)}$ is chosen as

$$\rho^{(j,k)} = \begin{cases} \eta^{(j,k)}(t), & \text{if} \quad \eta^{(j,k)}(t) < \eta^{(j,k)}_{\text{crit}} \\ 1 - e^{-\sigma\eta^{(j,k)}(t)}, & \text{otherwise} \end{cases} \tag{4.17}$$

where $\eta^{(j,k)}_{\text{crit}}$ is a critical occupancy level in each road that the algorithm aims at keeping each edge below, and σ is a tuning parameter determining the value of $\rho^{(j,k)}$ when the occupancy of road (v_j, v_k) reaches this threshold. The choice of σ tunes the difference in the cost function when a road is above or below $\eta^{(j,k)}_{\text{crit}}$. We have chosen an exponential decay function because it emphasises a low cost of roads with a low occupancy, and a disproportionately higher cost for roads which already have a high occupancy.

We were motivated to design the cost function in this way in order to limit the inflow of vehicles to intersections with long or growing queues, which is observed in traffic flow studies to reduce the probability of gridlock [63].

The **Decentralised Routing Algorithm (DRA)** resulting from the combination of the two sensory functions given above, attempts to control traffic flow using decentralised rules applied at the level of the individual vehicle.

In contrast to a user simply entering their vehicle and following a route which takes into account traffic conditions at the beginning of their journey, in our methodology the user plans their route in real time during the journey. When a large enough percentage of vehicles engage in this method, the resultant effect is load balancing of traffic on the network.

The measurement of data and communication to vehicles is distributed across the junctions. The junctions deal with one-way communication with the vehicles closest to it, providing the occupancy level of any connected roads (information which would be a few bytes in size). This contrasts with a system which might update all vehicles with the real-time occupancy of roads in the whole network, which would have a much higher communications overhead.

The computation of routes is done by the vehicles themselves, using the occupancy data received from the next intersection. The vehicle already has knowledge of its intended destination, the network topology, and of free-flow travel times. This distributes the computational overhead amongst the vehicles themselves, removing the necessity for a central controller whilst still enabling the vehicles to behave in a cooperative manner

Note that our solution to the vehicle routing problem is highly decentralised; Fig. 4.1 shows how vehicles only require one-way communication with local intersection controllers in order to receive information on local traffic conditions. The global information on the network is stored by the vehicles and does not change during the journey, while local information about road conditions in the immediate vicinity is updated at the nearest intersection. We also note that in our implementation of the algorithm we stopped vehicles

[1]Congestion here is considered the reduction of flow through the network. For a single road this relates to the density of traffic going above a certain value according to the Lighthill-Whitman model. Occupancy and density are related measures, in that occupancy is a dimensionless measure of vehicle density. Therefore, we allow for congestion by measuring occupancy and attempting to adjust the cost of a route in order to balance loads across the network

from taking roads if the shortest path would then take them along the same road in the opposite direction, causing unnecessary loop planning from the vehicles.

4.3 Tuning the Parameter α

Table 4.2: Optimal Values of α from Fig. 4.2 and Fig.4.3

Network Topology	Best Performing Value(s) of α	
	without Traffic Lights	with Traffic Lights
10x10 Lattice	{0.95}	{0.95}
Spiderweb	{0.85, 0.9}	{0.9}
Random	{0.65, 0.7}	{0.95}
Small-world (10x10)	{0.4}	{0.8}

To investigate the performance of the algorithm and the importance of tuning the parameter α, we used the well known traffic simulator SUMO (Simulation of Urban Mobility) [12]. Using the SUMO API (TraCI), several Python modules were produced in order to allow vehicles to be rerouted at each junction, according to the selected routing algorithm.

For all simulations, journeys were generated randomly using the 'randomTrips.py' script (included within the SUMO tools library).

The aim is to investigate numerically:

- The existence of an optimal range of values of α, which minimises unacceptable delays.
- The performance of the DRA against routing based on the shortest path only.
- The effect of different road network structures on the performance of the algorithm and hence on their resilience to congestion.

Table 4.1: Network Properties

Network Topology	**Nodes**	**Edges**	**Mean Degree**	**Network Diameter**
10x10 lattice	100	180	1.8	1800
Spiderweb	100	190	1.9	2000
Random	100	180	1.8	1364
Small-world (10x10)	110	190	1.7	1371
32x32 lattice	1152	2112	2	6200
Small-world (32x32)	1152	2112	2	5839

- The effect of the market penetration rate of the DRA (i.e. an assessment of the fraction of vehicles in the car that needs to adopt the proposed DRA in order for the network to become more resilient to congestion).

We used a selection of synthetic networks to explore how tuning the value of α affects the performance of our routing algorithm, and then compared the performance of the DRA to taking only the shortest path. As our networks are small, and therefore journeys are short, we used a high value of $\beta = 5$ in order to compare performance. The acceptable delay threshold therefore becomes $\hat{\omega} = 5 \cdot \bar{\mathcal{D}}$ for each network. This means that journeys can be delayed by up to 5 times the mean expected travel time in free flow conditions, but in real terms this is a delay will only be a few minutes, and it provides an equal measure by which to compare the algorithms tested in this study, regardless of the size of β. We performed simulations both in the presence of traffic lights and also in their absence, in order to determine how their presence affects the performance of the DRA. Finally, we scaled up the size of two of our networks to determine robustness to scale. We studied the performance of DRA on a selection of network topologies of different sizes (see Table 4.1 for their structural properties):

(a) 10x10 lattice

(b) Spiderweb network

(c) Random network created by rewiring a lattice network

(d) Small-world like network created by rewiring a lattice network

(e) 32x32 lattice

(f) Small-world like network created by rewiring the 32x32 lattice

The lattice networks analysed are inspired by road layouts in places such as Manhattan Island in New York City (Google Map - `http://tinyurl.com/kz32ut8`). Any vehicle wanting to travel diagonally across the network can choose among a large selection of shortest paths, thereby favouring a routing algorithm which considers all the possible paths.

The spiderweb network is a collection of 10 ring roads of increasing size, connected by another 10 roads which travel between the innermost and outermost rings. The spiderweb network is reminiscent of road structures around cities like Beijing (Google Map - `http://tinyurl.com/ojyexwk`). The spiderweb network can be formed by taking the 10x10 lattice and considering periodic boundary conditions so that the opposite sides of the lattice are touching. In so doing the horizontal roads in the lattice become the ring roads, and the vertical roads in the lattice become the connecting roads.

The random and small-world networks were generated by rewiring a lattice network, in order to generate networks with similar numbers of nodes and edges but with varying degree distributions. Links were removed from the edges and then reassigned with a probability related to their degree, in order to produce the desired degree distribution

(following a methodology for rewiring networks [79] with adaptions inspired by work on generating planar graphs [74]). There is evidence to suggest that a large number of urban road networks have a scale-free structure [59].

4.3.1 Validation of the Existence of an Optimal Value of α via Numerical Simulations

Simulations without Traffic Light Controlled Intersections

We investigate the dependence of $\bar{\omega}(\lambda)$, which we defined as the mean delay, on α. We find that for each network there is an optimal range of values of α which allow for a relatively larger value of the car generation rate λ and hence corresponds to a network which is more resilient to the onset of congestion. Specifically, we refer to $\bar{\omega}(\lambda, \alpha)$ as the delay observed, and we seek values of α such that,

$$\bar{\omega}(\alpha, \lambda) \leq \hat{\omega} \tag{4.18}$$

As $\hat{\omega}$ is constant equation 4.18 defines implicitly in the (λ, α) plane the critical curve of values of λ, say $\hat{\lambda}(\alpha)$, over which congestion is detected.

We plot this critical curve numerically for different network configurations in Fig. 4.2. Here we notice that such a curve possesses local and global maxima as a function of α suggesting that some choices of this parameter lead to more resilient networks and better performance. The values of α for which $\hat{\lambda}$ is maximal are reported in Table 4.2 (including the case when traffic lights are present).

These figures show the results of approximately 900 simulations per network. Specifically, each point in the Figure is obtained by evaluating the mean delay, $\bar{\omega}$, for fixed values of α and λ.

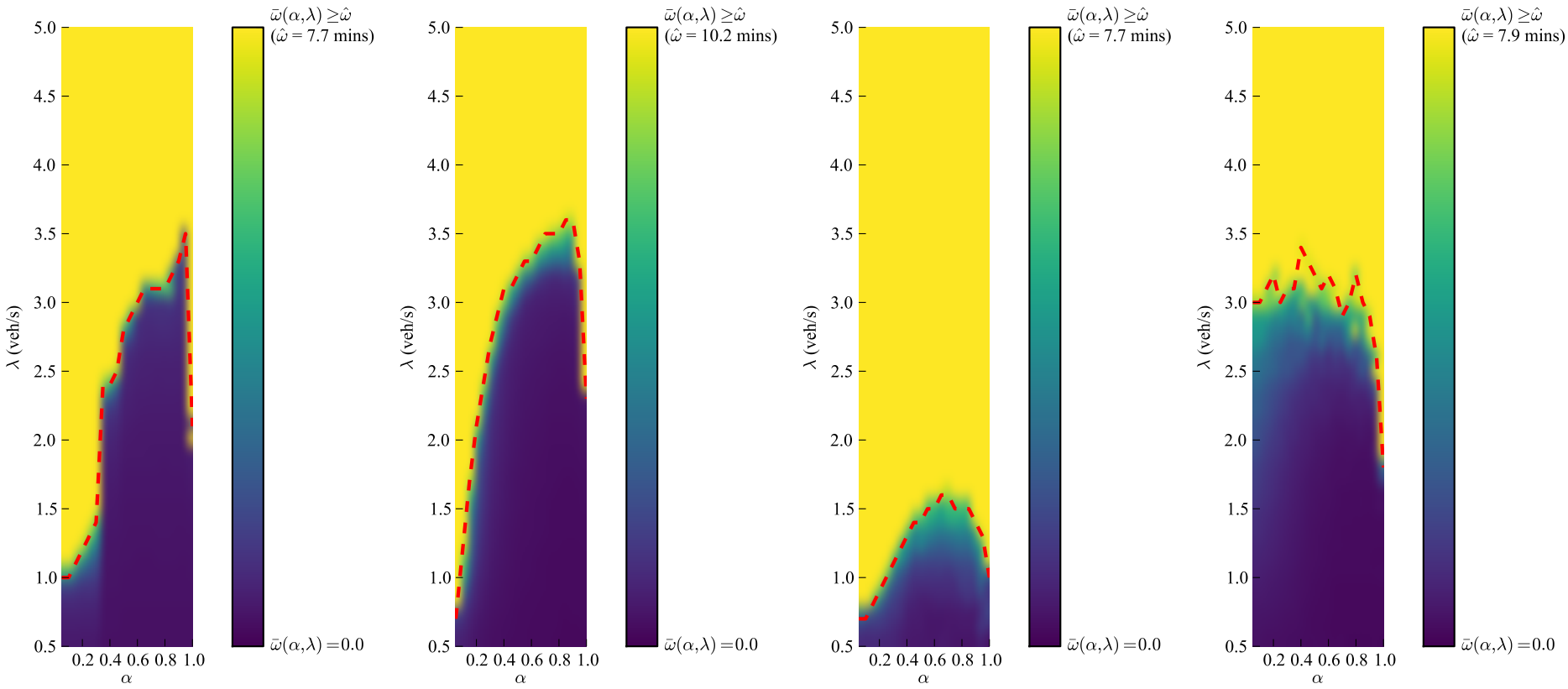
Setting $\alpha = 0$ represents routing based entirely on avoiding congestion, and so we would expect a poor performance because the vehicle simply avoids congested roads rather than moving towards its destination, so simulations were not performed with this value. Setting $\alpha = 1$ represents routing based entirely on the shortest path (and in the case of multiple shortest paths, one will be chosen at random).

In the lattice (Fig. 4.2a) we observe that the optimal value of α is approximately equal to 0.95, meaning that better performance is obtained by giving more weight to following the shortest path than to avoiding roads with a high occupancy. However, when $\alpha = 1$ we find that $\hat{\lambda}$ is lower than when $0 < \alpha < 1$, so we see that it is important to choose an intermediate value that weights some of the cost of taking high occupancy roads.

The spiderweb network (Fig. 4.2b), shows similarities to the 10x10 lattice, as we observe that choosing α in the range (0.85,0.9) gives better performance than for other values of α.

In the random network (Fig. 4.2c) we observe again a global maximum but at a different value of α which is around $\alpha = 0.65$. Furthermore, we observe that the performance of the DRA is reduced in this network in comparison to the lattice and spiderweb networks, in terms of the maximum observed value of $\hat{\lambda}(\alpha)$.

In the small-world network (Fig. 4.2d) we observe multiple local maxima with the global maximum corresponding to a value of α approximately equal to 0.4. As such this is the only network in which the weighting of α is more biased towards the occupancy based cost function.



(a) 10x10 Lattice (b) Spiderweb (c) Random (d) Small-world (10x10)

Figure 4.2: Change in $\bar{\omega}(\alpha, \lambda)$ (mean delay, measured in minutes) as α (the DRA tuning parameter) and λ (the car generation rate, measured in vehicles per second) are varied for each network, **without traffic lights**. Dark blue represents regions where delay was below the threshold $\hat{\omega}$, and yellow where it was above. The dashed red line ($\hat{\lambda} = \hat{\lambda}(\alpha)$) indicates the relationship between α and the maximum value of λ reached before $\bar{\omega}(\alpha, \lambda)$ becomes greater than $\hat{\omega}$ (the acceptable delay threshold).

Our simulations clearly show the fundamental effect that the network topology and structure has on the performance of the routing algorithm. Nevertheless, in all the topologies being investigated we notice the beneficial effects of weighing both distance from destination and congestion on the roads when making routing choices. In particular, the simulations confirm the existence of an optimal range of values for the tuning parameter α in the cost function that makes our algorithm adaptable to different network structures and definitely better performing than shortest path routing.

The next step is to assess the effects of traffic lights and whether their presence changes the scenarios uncovered so far.

Simulations with Traffic Light Controlled Intersections

We repeated the simulations with traffic lights included in the networks. Traffic lights were added by SUMO using inbuilt options for building the network, which included SUMO guessing appropriate timings. The results of the simulations for different network structures is depicted in Fig. 4.3.

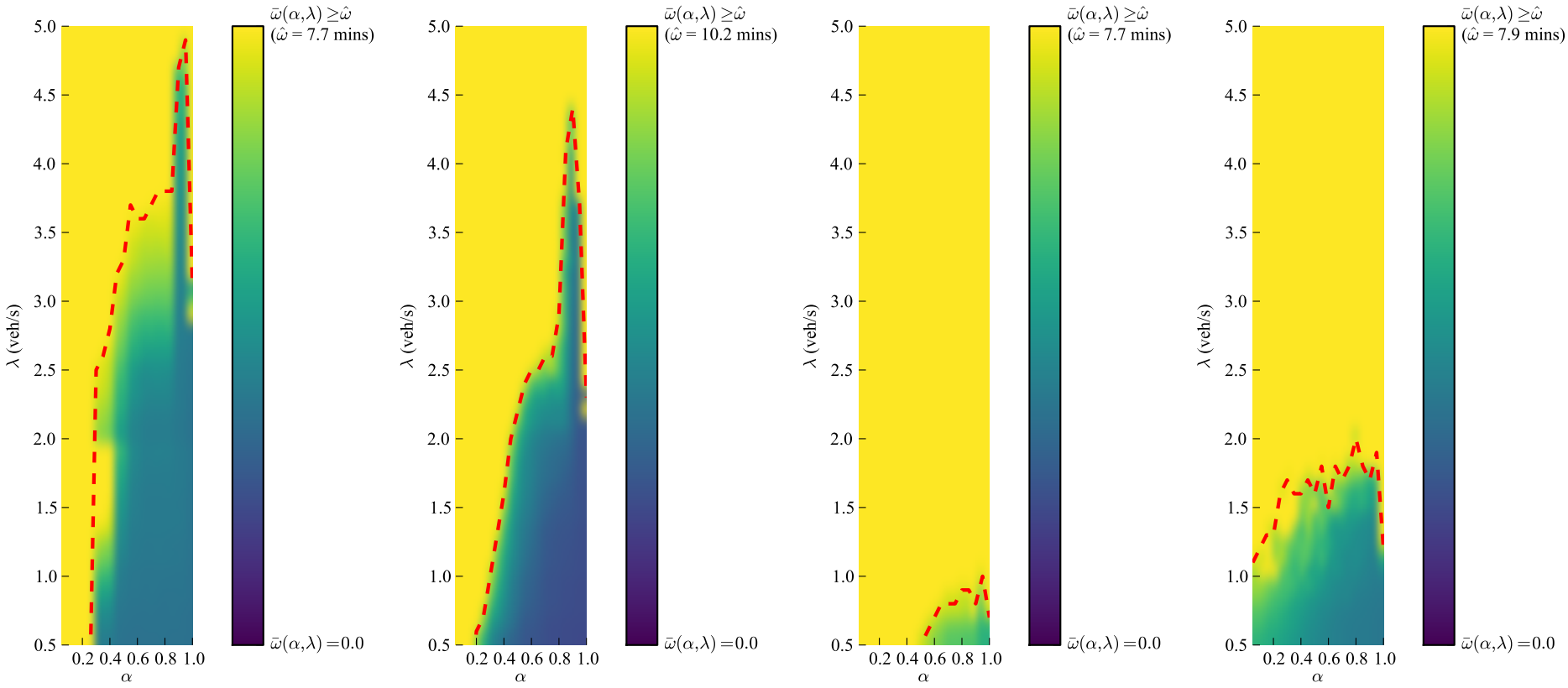
It is immediately apparent by contrasting Fig. 4.2 and Fig. 4.3 that traffic lights do have a significant effect on the performance of the routing algorithm. Surprisingly this can be beneficial as in the case of lattice and spiderweb networks or not as in the case of random and small-world networks. In any case, we notice again the presence of both local and global maxima and the fundamental effect that the network structure has on determining the performance.

More specifically, in the lattice network (Fig. 4.3a) we observe that the best value for α is still 0.95, as in the network without traffic lights.

We also observe that when $\alpha < 0.5$ the DRA performs better in the lattice without traffic lights than compared to the lattice with traffic lights; however, when comparing the maximum values of $\hat{\lambda}(\alpha)$ achieved in each scenario, the value is higher in the network with traffic lights.

In the spiderweb network (Fig. 4.3b) we make similar observations to those in the lattice network. We find that traffic lights do not affect the presence or location of the maximum which is still located at approximately 0.9. We find that $\hat{\lambda}(\alpha)$ is lower in the network with traffic lights when $\alpha \leq 0.7$, but that when $0.7 < \alpha \leq 1$ the values of $\hat{\lambda}(\alpha)$ are greater, and the network resilience to congestion is higher when traffic lights are present.

In the random network (Fig. 4.3c) we observe that the best choice for α is to set $\alpha \approx 0.95$. $\hat{\lambda}(\alpha)$ shows the same trend seen in the lattice and spiderweb networks, whereby higher values of α (excluding $\alpha = 1$) outperform lower values when traffic lights are present. However, we find in the random network that the value of $\hat{\lambda}(\alpha)$ is generally lower when traffic lights are present, which contrasts with our findings in both the lattice and spiderweb networks.



(a) 10x10 Lattice (b) Spiderweb (c) Random (d) Small-world (10x10)

Figure 4.3: Change in $\bar{\omega}(\alpha, \lambda)$ (mean delay, measured in minutes) as α (the DRA tuning parameter) and λ (the car generation rate, measured in vehicles per second) are varied for each network, **with traffic lights**. Dark blue represents regions where delay was below the threshold $\hat{\omega}$, and yellow where it was above. The dashed red line ($\hat{\lambda} = \hat{\lambda}(\alpha)$) indicates the relationship between α and the maximum value of λ reached before $\bar{\omega}(\alpha, \lambda)$ becomes greater than $\hat{\omega}$ (the acceptable delay threshold).

In the small-world network (Fig. 4.3d) we see that a shift in the optimal value of α from 0.4 to 0.8, meaning that this trend of shifting maxima in the presence of traffic lights is seen across all of the networks studied here. We also find that in the small-world network the maximal value of $\hat{\lambda}(\alpha)$ decreases when traffic lights are introduced, contrasting with the lattice and spiderweb networks but matching findings in the random network.

Comparing the results in the networks with and with traffic lights (see Table 4.2) we observe that the optimal values of α increase in the networks with traffic lights when compared to those without. We can explain this behaviour of the DRA due to the relationship between α, the behaviour of individual vehicles, and the effect of traffic lights on travel time. A vehicle has a certain probability of being delayed by every traffic light that it encounters, keeping it at a junction for a period of time. When α is small the cars are encouraged to take detours and therefore travel through a larger number of junctions, increasing the number of traffic lights they come into contact with. This increases the probability of encountering additional delays on their journey, and hence the performance of the DRA is decreased.

In the case of the lattice and spiderweb networks where we find that the DRA performs better when traffic lights are present, for high values of α, we believe that the traffic lights are able to dissipate queues evenly across the network, but that this is only effective when traffic is spread evenly amongst the roads. This is more feasible in networks with many alternative paths to a destination.

Therefore, in networks with traffic lights, we conclude that it is preferable to choose higher values of α than in networks without, however, we still find that $\alpha = 1$ (i.e. routing based solely on shortest distance) is not optimal.

4.3.2 Estimation of the Optimal Values of α

We present an analysis of the optimal values of the parameter α in each network. This analysis also allows us to better understand observations from numerical simulations.

Properties of the Chosen Cost Functions

We consider the effect of varying α on the likelihood of a vehicle taking an alternative route to the shortest path. We consider the comparison of two paths, which was introduced in section 4.2.1.

Consider a choice between two alternative roads for the i-th vehicle. Road 2 will take us on the fastest path to the destination if the network has no congestion, and road 1 will take us on an alternative path which is equal to or longer than the fastest path (potentially avoiding some congestion). For simplicity, we denote the two possible roads in our equations by the subscripts 1 and 2 (as opposed to the convention of using the pair of end nodes denoted by (v_j, v_k)), and do away with the i subscript for this section. We will, therefore, refer to the distance and occupancy based cost functions for the two roads as ϕ_1, ϕ_2, ρ_1 and ρ_2.

We wish to control the behaviour of the vehicle and determine when it will accept a longer travel time in order to avoid a congested road. We showed previously that we can

determine the maximum value of α which will cause a vehicle to detour onto an alternate path if we are able to evaluate (4.14) and (4.15). In order to evaluate equation (4.15) we need to find expressions for $\phi_1 - \phi_2$ and $\rho_2 - \rho_1$.

Looking at equation (4.16), used for calculating the "travel-time" based cost function, we can see that the denominator is a constant for a given network, and is at least as great as the travel time for the longest path in the network The numerator is the expected travel time to the destination of the i-th vehicle taking road (v_j, v_k). The numerator, therefore, increases or decreases with an increasing or decreasing expected path length when taking the road in question.

We can rewrite equation (4.16) as,

$$\phi_i^{(v_j,v_k)} = \frac{l_i^{(v_j,v_k)}}{\bar{L}}, \quad d_n \leq \bar{L} \tag{4.19}$$

where $l_i^{(v_j,v_k)}$ is the length of the path which will be taken by the i-th vehicle, if it takes road (v_j, v_k),

$$l_i^{(v_j,v_k)} = \mathcal{D}(v_j, v_k) + \mathcal{D}(v_k, d_i) \tag{4.20}$$

and $\bar{L}$ is the maximum possible path length through the network,

$$\bar{L} = \max_{v_s,v_t \in \mathbf{V}} \mathcal{D}(v_s, v_t) + \max_{(s,t)\in \mathbf{E}} \mathcal{D}(v_s, v_t) \tag{4.21}$$

Definitions for these symbols can be found with equation (4.16).

Given that in equation (4.19) $\bar{L}$ is constant for a given network, then the value of $\phi_i^{(v_j,v_k)}$ depends solely on $l_i^{(v_j,v_k)}$, which becomes greater the longer the travel time to the destination. Therefore, in this case $\phi_2 < \phi_1$, which we can rewrite to say that $\phi_1 = \Delta \cdot \phi_2$, $\bar{c} \geq 1$, where Δ is the ratio of the travel times for the two routes.

Given that no path can be longer than the longest path (i.e. $\bar{L} > l_i^{(v_j,v_k)}$ always), then $\phi_1, \phi_2 \leq \bar{L}$ and we can show that the maximum value of $\phi_1 - \phi_2$ for any value of Δ is found by,

$$\phi_1 - \phi_2 = 1 - \frac{1}{\Delta} \tag{4.22}$$

We plot this in Figure 4.4a. In this figure we compare the ratio of the lengths between the two routes (Δ) with the maximum difference in their travel time-based cost functions. We see that the difference $\phi_1 - \phi_2$ increases non-linearly with the length of the longer path, thus short detours will be more readily accepted than long detours. For completeness we plot Figure 4.4b, which shows the value of $\phi_1 - \phi_2$ for all values of ϕ_2 and a small range of Δ. This figure shows that the value $\phi_1 - \phi_2$ will increase linearly as ϕ_2 increases, but increase non-linearly as Δ increases (i.e. the higher the travel time of the shortest path, the higher the effect of Δ on increasing $\phi_1 - \phi_2$).

We can use a similar approach to visualise the change in the value of $\rho_2 - \rho_1$ as the occupancy of the roads 1 and 2 are varied. We have already shown that high values of α make vehicles less likely to avoid high occupancy roads, and low values of α do the

opposite, but we wish to quantify in exactly which situations the vehicle will choose to avoid the road with higher occupancy. We firstly plot our occupancy based cost function $\rho^{(v_j,v_k)}$ as a function of percentage road occupancy, and the tunable parameters σ and η_{crit}. We show several cases in Figure 4.6a. We plot the the case when $\sigma = 10$ and $\eta_{\text{crit}} = 0.2$ (20% occupancy) and show the change in $\rho_2 - \rho_1$ as road occupancy varies in Figure 4.6b. We observe in this figure that when the occupancy of the road on the fastest path to the destination (road 2) goes above the critical occupancy, and the occupancy of the road on the alternative longer route (road 1) stays below the critical occupancy, we see an immediate jump in $\rho_2 - \rho_1$ from ≈ 0.2 to ≈ 0.7. This indicates a switching behaviour, which we can use to control when a vehicle will detour from the fastest path onto an alternative.

Estimation Method

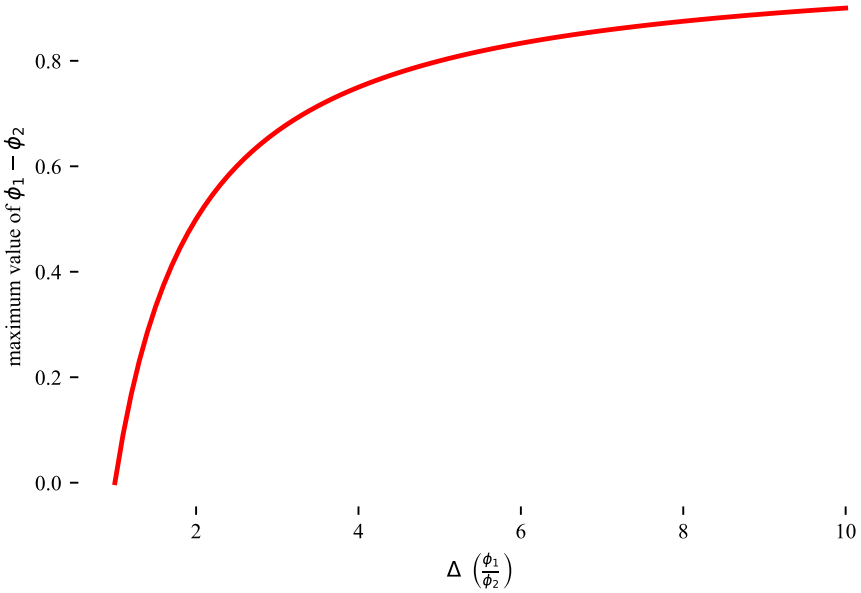
In Figure 4.7 we visualise how α can be chosen to control when a vehicle will choose between road 1 and road 2. We plot $\phi_1 - \phi_2$ (x-axis) against $\rho_2 - \rho_1$ (y-axis) and the minimum value of α which will cause the vehicle to reroute to the road on the longer path (z-axis).

Firstly we consider the possible ranges values of $\phi_2 - \phi_1$ which will be encountered by our vehicle. We noted previously that in Figure 4.6b the value $\rho_2 - \rho_1$ will jump from 0.2 to 0.7 when the road on the shortest path (road 2) goes above its critical occupancy. We have greyed out the region where $0.2 < \rho_2 - \rho_1 < 0.7$ in Figure 4.7, as we want to focus on behaviour outside of this region.

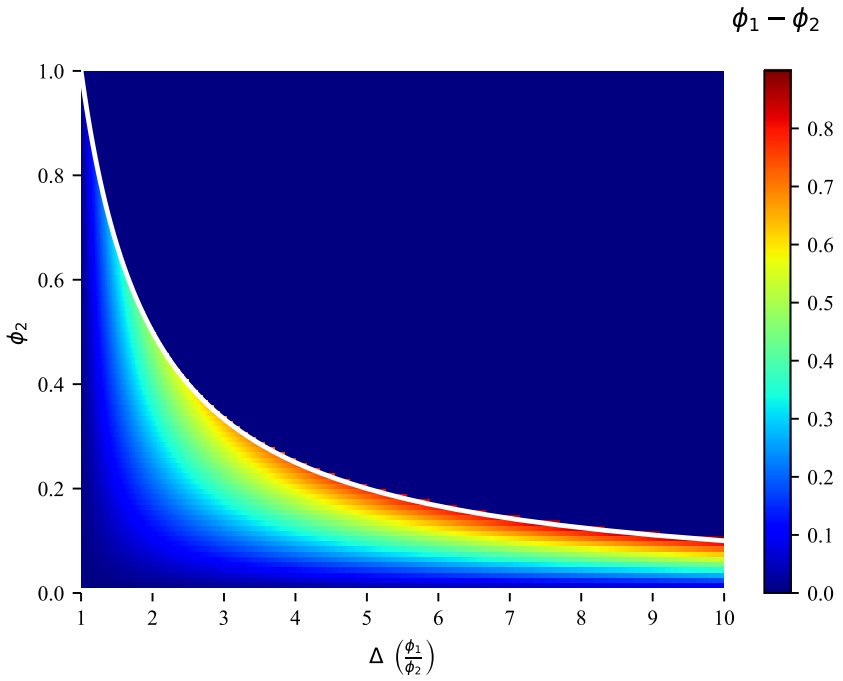
In Figure 4.5 we have information for each network about the range of values for $\phi_1 - \phi_2$ which will be encountered in the network, the relative length of a detour to the shortest path at that value (Δ), and the frequency we encounter these values. We also know from Figure 4.4b that the length of a detour increases with both $\phi_1 - \phi_2$ and Δ. We can, therefore, predict how tuning α will change when an alternative path is followed (in our example by taking road 1) over a shorter path (in our example by taking road 2) for certain values of $\phi_1 - \phi_2$ and Δ.

In Figure 4.5a we see that in the lattice network there are only a few possible values for $\phi_1 - \phi_2$, in the network those values are approximately 0, 0.1, and 0.2. We therefore expect to find our optimal value of α by considering the region of Figure 4.7 where $\phi_1 - \phi_2 < 0.2$. We found through simulation that the delay to vehicles in the network was lowest when $\alpha = 0.95$, which in Figure 4.7 is shown by the dark red line contour line. When $\alpha = 0.95$ a vehicle will only take road 1 over road 2 when $\phi_1 - \phi_2$ is slightly greater than 0.1, which tells us that only a certain proportion of possible detours (those up to $\Delta = 3$) are possible. Furthermore, when both roads are above 20% occupancy, or both are below 20% occupancy, then vehicles will only take route 1 when $\phi_1 - \phi_2 = 0$ (i.e. when both paths have the same travel time).

In Figure 4.5b we observe that in the spiderweb network there is a much wider range of values for both $\phi_1 - \phi_2$ and Δ. The most frequent values to occur are when $\Delta \approx 1.5$ and $\phi_1 - \phi_2 < 0.1$, but the most extreme occurrences go up to $\Delta = 14.5$ and $\phi_1 - \phi_2 = 0.4$. In

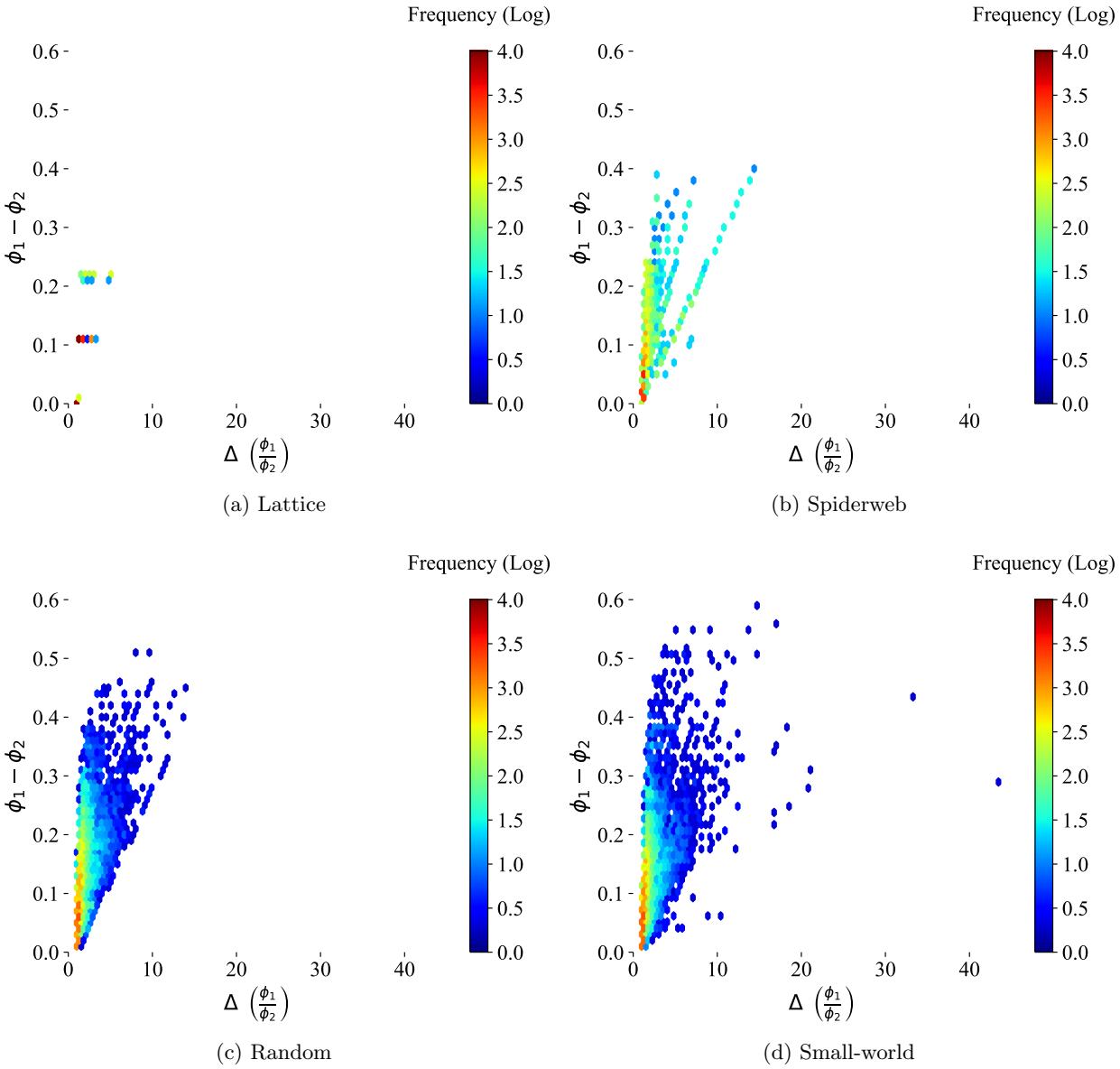


(a) Maximum value of $\phi_1 - \phi_2$ as the ratio between them (Δ) increases. The relationship is non-linear, suggesting that detours which have a higher Δ will require a lower value of the tuning parameter α to be taken by a vehicle.



(b) Actual value of $\phi_1 - \phi_2$ as the ratio between them (Δ) and the travel time cost of the shortest path (ϕ_2) increases. The figure shows that the difference in costs will increase linearly as the length of the shorter of the two paths increases, and non-linearly as the ratio between the two paths increases.

Figure 4.4: Relationship between the difference in the travel time based cost functions of roads 1 and 2 ($\phi_1 - \phi_2$), and the ratio of the longer of the proposed routes to the shorter (Δ).



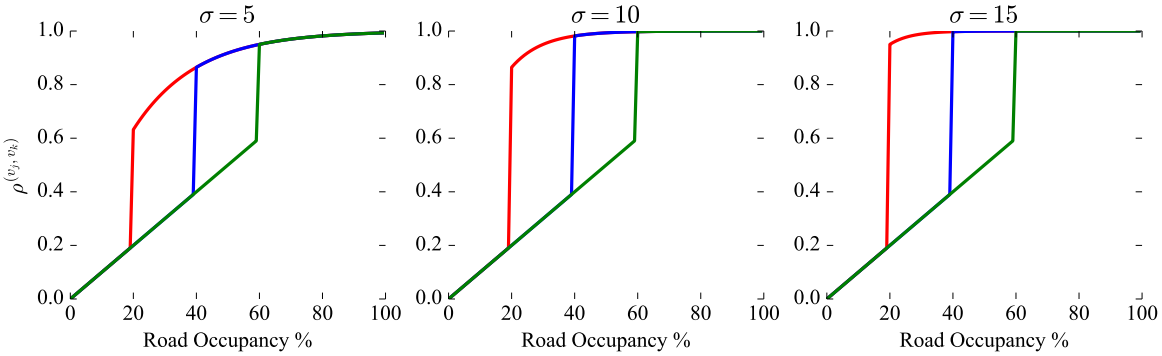
(a) Lattice

(b) Spiderweb

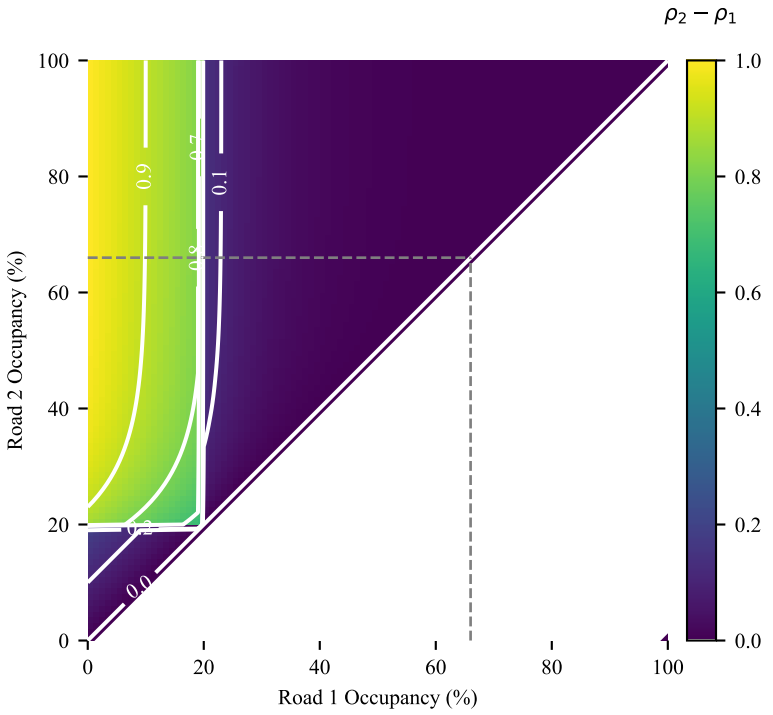
(c) Random

(d) Small-world

Figure 4.5: Calculation of $\phi_1 - \phi_2$ for all road choice comparisons in each of the synthetic networks, plotted against the ratio $\Delta\left(\frac{\phi_1}{\phi_2}\right)$ with the frequency of occurance in the network indicated by the colour scale (the colour scale is logarithmic). The figures show for each network the relative frequency of detours which increase travel time by a factor of Δ, and the corresponding value of $\phi_1 - \phi_2$. We can use this information to determine the values of $\phi_1 - \phi_2$ we expect to encounter in the network when tuning the parameter α.



(a) Value of the cost function $\rho^{(v_j, v_k)}$ as road occupancy varies. The occupancy of road 2 is greater than or equal to that of road 1. Three figures are given, showing the effect of varying the tunable parameter σ. The lines in each figure indicate the chosen critical occupancy (η_{crit}) as 0.2 (red), 0.4 (blue), or 0.6 (green).



(b) Evaluation of $\rho_2 - \rho_1$ as a function of the occupancy of roads 1 and 2. The tunable parameters have been set at $\sigma = 10$, $\eta_{\text{crit}} = 0.2$, as in the numerical simulations. We overlay a contour plot to highlight what happens when the road which will take us on the fastest route (road 2) goes above the critical occupancy of 20%. We find that so long as the alternative road (road 1) is below 20% occupancy, the value of $\rho_2 - \rho_1$ goes immediately from ≈ 0.2 to ≈ 0.7. The grey dash lines show the approximately the maximum range of occupancies seen during simulations (approx. 66%), due to the gap left between vehicles which prevents a road ever reaching 100% occupancy.

Figure 4.6: Relationship between the difference in occupancy based cost functions of roads 1 and 2 ($\rho_2 - \rho_1$), the tunable parameters σ and η_{crit}, and the % occupancy of each road (η).

our numerical simulations we saw the lowest vehicle delays when α was equal to 0.85 (or 0.9 in the presence of traffic lights). In Figure 4.7 we see that this means our algorithm will not take the road on the slower path when $\phi_1 - \phi_2 > 0.2$ and from Figure 4.5b this limits the maximum value of Δ to approximately 8.

In Figure 4.5c we observe the possible values of $\phi_1 - \phi_2$ in the random network. In the random network the maximum value of $\phi_1 - \phi_2$ is greater than either the lattice or spiderweb networks at slightly over 0.5. In our simulations without traffic lights the optimal value of α was around 0.65 or 0.7, in the random network, which corresponds to a maximum value of $\phi_1 - \phi_2 \approx 0.4$ and $\phi_1 - \phi_2 \approx 0.5$ respectively when we look at Figure 4.7. This means that all detours were potentially viable, with a maximum Δ of 14, when the algorithm was performing optimally with respect to reduced trip delays. However, at these values of α we also would find that the value of $\rho_2 - \rho_1$ becomes much more influential on how long the detour can be, for example when $\alpha = 0.7$ and $\rho_2 - \rho_1 = 0.7$, only detours where $\phi_1 = \phi_2 < 0.3$ will be considered.

When traffic lights are introduced to the random network we see the optimal value of α increase to 0.95. As traffic lights increase delays, and longer detours increase the chance of encountering a red light, this increase in α will mean that vehicles only detour when $\phi_1 - \phi_2 < 0.1$ and $\Delta < 4.5$.

In Figure 4.5d we find that the small-world network has the largest distribution of $\phi_1 - \phi_2$ and Δ in all the networks. The maximum value of $\phi_1 - \phi_2$ is approximately 0.6, and the maximum value of Δ is 44, however, these values are spread far from the most frequent values. The most frequent values in the small-world network are similar to those in the random network, when $\phi_1 - \phi_2 < 0.1$ and $\Delta \leq 1.5$. In simulations without traffic lights we found that the small-world network was difficult to tune, however, the optimal value of α from our results was 0.4. In Figure 4.7 we see that this value of α will cause a vehicle to always take the alternative route when the value of $\rho_2 - \rho_1$ is greater than 0.7. This means that all detours could possibly be taken. The introduction of traffic lights (and associated delays) increase the optimal value of α to 0.8, which would reduce the maximum value of $\phi_1 - \phi_2$ considered for a detour to 0.25. Interestingly this value allows for the most frequently occurring lengths of detour to be allowed (up to $\Delta \approx 20$), but just excludes the largest detour in the network of $\Delta = 44$.

Analysis of the cost functions and their relationship to α for each network indicates that the optimal value of α is mostly dependent upon controlling the relative length of detours a vehicle can take. However, in all of the networks it appears that an important correlation is for the largest cluster of detours (i.e. those regions which appear red in Figure 4.5 to be permissible when there is a large difference in occupancies between the two roads, but for large or extreme values of Δ to be filtered out.

4.4 Comparison with Dijkstra's Algorithm

Next, we compare our routing algorithm against routing using only the shortest path, as calculated using Dijkstra's algorithm [37]. The edge weights used in this calculation are the free-flow travel times.

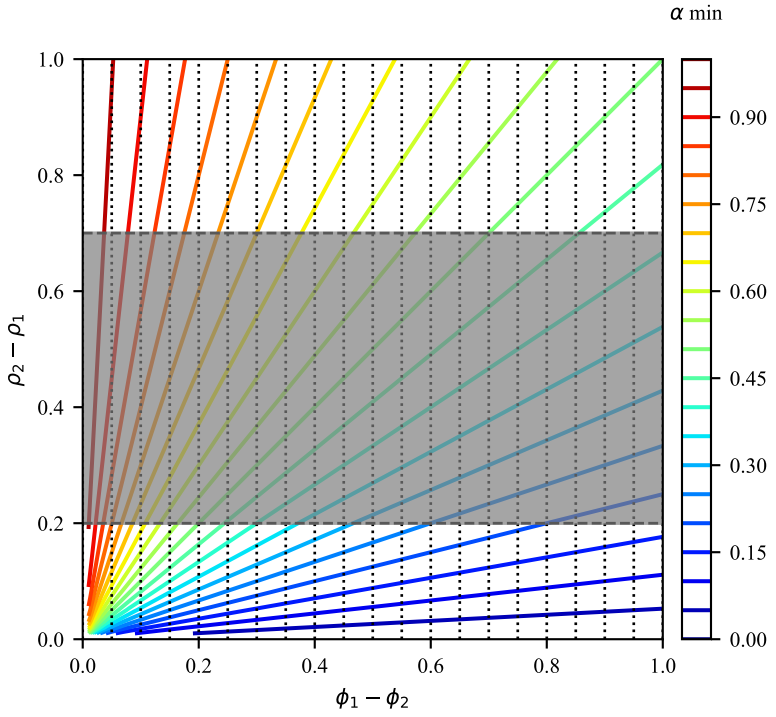
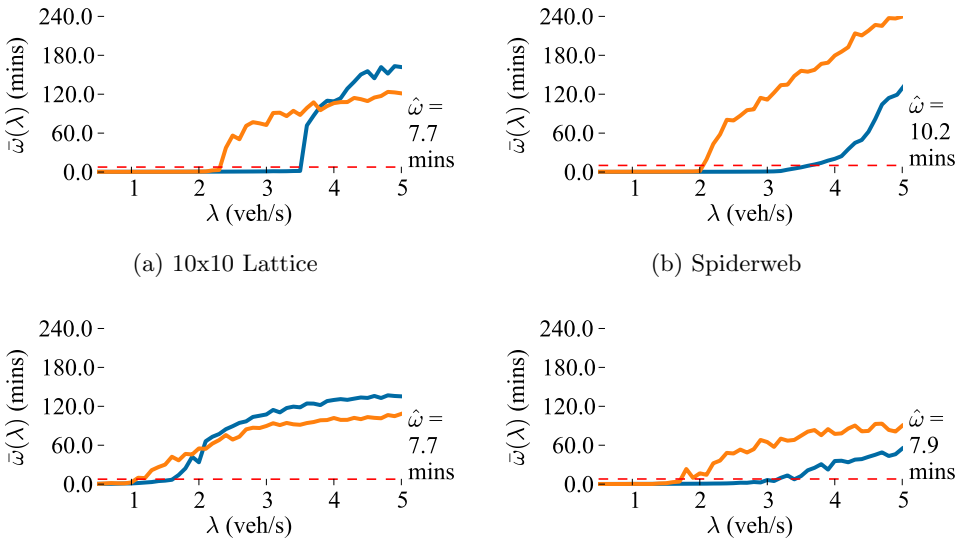


Figure 4.7: Contour plot of the minimum value of α (z-axis) that will ensure that the i-th vehicle takes the road on the fastest route (road 2) over the less congested alternative road (road 1). As the difference in the expected travel time costs of the roads ($\phi_1 - \phi_2$) increases, this value of α decreases, and as the difference in occupancy costs ($\rho_2 - \rho_1$) increases the value of α increases. The greyed out region indicates values of $\rho_2 - \rho_1$ which do not occur with our chosen cost function. Given a specific value of α, we can see from this figure under what conditions a vehicle will reroute.

Table 4.3: Validation of DRA Compared to the Shortest Path (Dijkstra)

Network Topology	$\hat{\lambda}$ without Traffic Lights		$\hat{\lambda}$ with Traffic Lights	
	Dijkstra	DRA	Dijkstra	DRA
10x10 Lattice	2.3	3.5 (+52%)	2.2	4.9 (+123%)
Spiderweb	2.0	3.6 (+80%)	2.6	4.4 (+69%)
Random	1.0	1.6 (+60%)	0.7	1.0 (+43%)
Small-world (10x10)	1.6	3.4 (+127%)	1.1	2.0 (+82%)



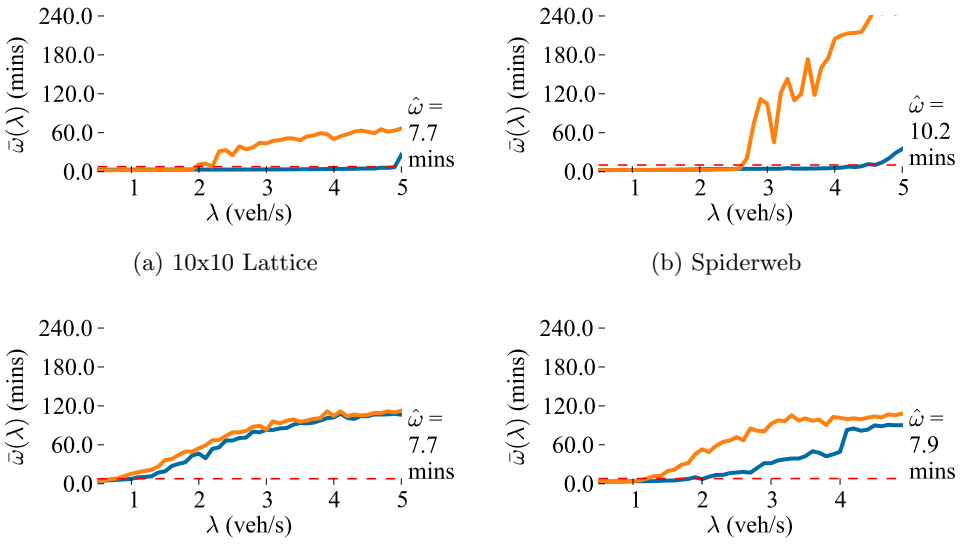
(a) 10x10 Lattice

(b) Spiderweb

(c) Random

(d) Small-world (10x10)

Figure 4.8: Change in $\bar{\omega}(\lambda)$ (mean delay, measured in minutes) when varying λ (car generation rate, measure in vehicles per second) in networks without traffic lights, when routing using the shortest path (orange line) and the DRA (blue line - using the best performing value of α for each network (see Table 4.2)). $\hat{\omega}$ (the acceptable delay threshold) is shown by the dashed red line for each network.



(a) 10x10 Lattice

(b) Spiderweb

(c) Random

(d) Small-world (10x10)

Figure 4.9: Change in $\bar{\omega}(\lambda)$ (mean delay, measured in minutes) when varying λ (car generation rate, measure in vehicles per second) in networks with traffic lights, when routing using the shortest path (orange line) and the DRA (blue line - using the best performing value of α for each network (see Table 4.2)). $\hat{\omega}$ (the acceptable delay threshold) is shown by the dashed red line for each network.

4.4.1 Simulations without Traffic Light Controlled Intersections

We start by considering the case without traffic lights at the intersections. Fig. 4.8 shows the results of the simulations for DRA and Dijkstra for each of the network structures considered in this chapter. In the figure, we plot the average delay $\bar{\omega}(\lambda)$ (in minutes) against λ (in vehicles per second), and observe the critical value $\hat{\lambda}$ of the car generation rate above which the average delay becomes greater than the congestion threshold $\hat{\omega}$ defined earlier for both shortest path routing and the DRA, where the control parameter α has been tuned in the optimal regions for each network structure which were highlighted in the previous section.

In the lattice (Fig. 4.8a), we find $\hat{\lambda} = 2.3$ using shortest path routing. In contrast, the DRA keeps $\bar{\omega}(\lambda)$ below $\hat{\omega}$ up to a maximum of $\hat{\lambda} = 3.5$, which represents an increase of over 65% with respect to shortest path routing.

In the spiderweb network (Fig. 4.8b) we find that both shortest path routing and the DRA exhibit similar performance to that seen in the lattice network. Again we notice a notable increase of the network resilience which can sustain car generation rates of up to 3.6 cars per time step in contrast to 2 cars per time step when the shortest path is used.

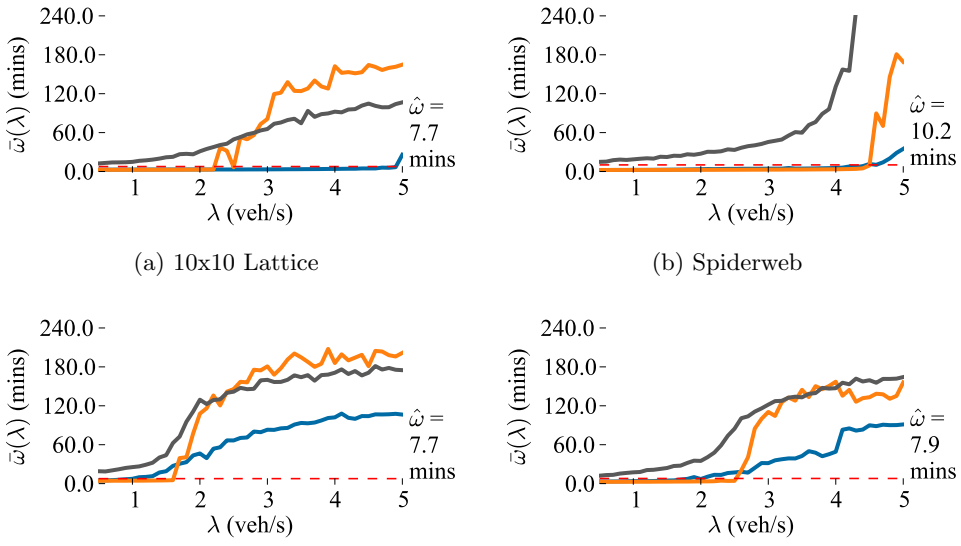
A notable increase of the network resilience is observed in the random network (Fig. 4.8c) where, despite performing the least well in comparison to the other topologies, the DRA guarantees a critical λ of approximately 1.6, higher than the threshold value for shortest routing. A similar increase in performance is also observed in the small world network (Fig. 4.8d) where shortest path routing enables $\hat{\lambda} = 1.6$, whereas the DRA was able to perform better again, increasing this value to $\hat{\lambda} = 3.4$.

In summation, the DRA outperforms routing using only the shortest path in all networks tested here, without traffic lights present. The increase in maximum car generation rate that the DRA exhibits over shortest path routing is 52% in the 10x10 lattice, 80% in the spiderweb network, 60% in the random network, and a surprising 127% in the small-world network.

4.4.2 Simulations with Traffic Light Controlled Intersections

Simulations were repeated with traffic lights present, and are reported in Fig. 4.9. Again we find that in general, the use of the DRA algorithm presented in this chapter improves performance and resilience of the network when contrasted to using only the shortest path. This is clear for the lattice (Fig. 4.9a) and the spiderweb networks (Fig. 4.9b) while in the random network (Fig. 4.9c) we find a smaller improvement (with the DRA still outperforming shortest path routing). As also shown for the small-world network (Fig. 4.9d), the presence of traffic lights in these networks reduces the increase in performance observed when traffic lights are not present.

Nevertheless, in terms of percentage improvements of the DRA over shortest path routing in these networks, we find that the DRA is able to increase $\hat{\lambda}$ by 123% in the lattice, 69% in the spiderweb, 43% in the random network. and 82% in the small-world network The results for both simulations with and without traffic lights are summarised in Table 4.3. We observe that in both sets of simulations, both the DRA and shortest



(a) 10x10 Lattice

(b) Spiderweb

(c) Random

(d) Small-world (10x10)

Figure 4.10: Figures showing change in $\bar{\omega}(\lambda)$ (mean delay, measured in minutes) when varying λ (car generation rate, measure in vehicles per second) in networks with traffic lights, when routing using DUA (orange line), LTTR (grey line) and the DRA (blue line - using the best performing value of α for each network (see Table 4.2)). $\hat{\omega}$ (the acceptable delay threshold) is shown by the dashed red line for each network.

Table 4.4: Validation of DRA Compared to DUA and LTTR

Network Topology	$\hat{\lambda}$ DUA	LTTR	DRA (% Diff. to DUA)
10x10 Lattice	2.2	None	4.9 (+123%)
Spiderweb	4.5	None	4.4 (-2%)
Random	1.6	None	1 (-38%)
Small-world (10x10)	2.5	None	2 (-20%)

path routing either benefit or suffer from the presence of traffic lights, with neither gaining an advantage over the other from their presence. However, in all simulations the DRA provides considerable improvement over routing using only the shortest path.

For the sake of completeness, we look next at the performance of other routing algorithms in order to further assess the performance of the DRA and the resilience of different network structures when DRA is used.

4.5 Comparison with Other Routing Algorithms

We compare the performance of our strategy against that of the following algorithms:

1. **Dynamic User Assignment (DUA)** calculated using methods presented in [41]. Vehicles are initially routed using the shortest path, and a simulation is then run using these routes. The condition of the road network is measured in order to detect the presence of congestion. The simulation is then re-run with a small percentage of

vehicles able to change their route to avoid the congested parts of the network. These steps are repeated until 50 iterations are complete, and allow a user equilibrium to be found. This routing strategy cannot be used in real-time but it provides a benchmark for comparison with our own algorithm.

2. **Live Travel Time Routing (LTTR)** which is implemented by allowing vehicles to change their routes every 15 seconds, using the current estimated travel times for elements in the network. This was done using the options available within SUMO itself. This simulates routing with information available from a service which provides high fidelity live travel times to drivers, and so can adjust their route ad-hoc.

The results of the simulations are shown in Fig. 4.10 for each of the network structures under consideration. We observe that apart from the random and small world networks where the DRA is outperformed by DUA, our algorithm shows better performance in all cases when compared to the LTTR. Specifically, in the lattice (Fig. 4.10a) we observe that the DRA outperforms both DUA and LTTR. DUA is able to keep $\bar{\omega} \leq \hat{\omega}$ up to $\lambda = 2.2$. The DRA compares favourably with its maximum value of $\hat{\lambda} = 4.9$. LTTR is unable to maintain acceptable delays, even at the lowest values of λ tested.

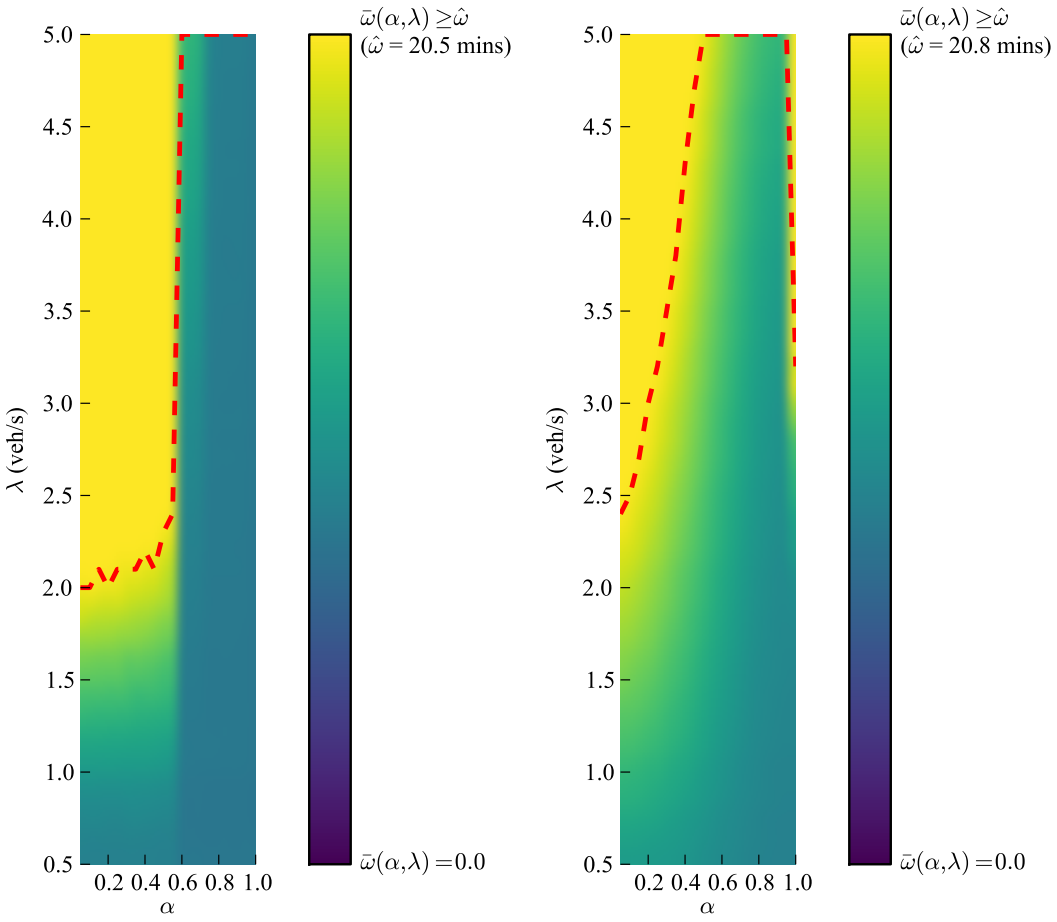
In the spiderweb network (Fig. 4.10b) we find that DUA and the DRA perform comparably, although DUA outperforms the DRA by a small margin, achieving $\hat{\lambda} = 4.5$ compared to $\hat{\lambda} = 4.4$. Once again LTTR is unable to maintain acceptable delays, even at the lowest values of λ tested.

In contrast to the lattice and spiderweb networks, we observe a significant difference in performance between the DRA and DUA when comparing results on the random network (Fig. 4.10c). DUA is able to achieve $\hat{\lambda} = 1.6$, whereas we find a critical value $\hat{\lambda} = 1$ when using the DRA. In this network, LTTR is still unable to maintain acceptable delays for any value of λ we tested.

Results from simulations in the small-world network (Fig. 4.10d) are similar to those seen in the random network. We observe that DUA outperforms the DRA by a small margin, maintaining reasonable delays so that $\hat{\lambda} = 2.5$ using DUA and $\hat{\lambda} = 2$ using the DRA. LTTR is unable to maintain acceptable delays for any value of λ tested.

We find that the best benchmark for performance is DUA, rather than LTTR, which performed poorly in all simulations, however it should be noted that DUA is a centralised and iterative routing method, requiring a prior knowledge of all vehicles and trips, whereas the DRA is able to route vehicles in real-time, using decentralised rules and only requiring local occupancy data at each intersection.

We find that the performance of the DRA compared to that of DUA varies according to the network topology. In the lattice network the DRA outperforms DUA, and in the spiderweb the performance is comparable, however, in the random and small-world networks, DUA outperforms the DRA by a considerable margin (when taking only $\hat{\lambda}$ into consideration). The percentage performance difference of the DRA compared with DUA was an increase of 123% in the lattice, a decrease of 2% in the spiderweb, a decrease of 38% in the random network, and a decrease of 20% in the small-world network. These results are summarised in Table 4.4.



(a) 32x32 Lattice

(b) Small-world (32x32)

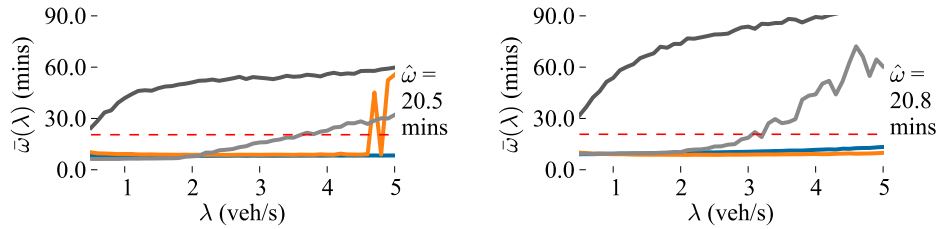
Figure 4.11: Change in $\bar{\omega}(\alpha, \lambda)$ (mean delay, measured in minutes) as α (the DRA tuning parameter) and λ (the car generation rate, measured in vehicles per second) are varied for each network, in large networks with traffic lights present. The dashed red line ($\hat{\lambda} = \hat{\lambda}(\alpha)$) indicates the relationship between α and the maximum value of λ reached before $\bar{\omega}(\alpha, \lambda)$ becomes greater than $\hat{\omega}$ (the acceptable delay threshold).

4.6 The Effect of Increasing Network Size

We generated larger networks in order to validate the performance of the DRA when the network size was increased. The networks tested are a 32x32 lattice, and a small-world network based on rewiring the 32x32 lattice, both of which contain traffic light controlled intersections.

The results of the simulations are shown in Fig. 4.11. We notice that, as in the case of their smaller counterparts, these networks show better resilience for certain values of the control parameter α in the cost function at the core of our DRA with the results being comparable to those already described for small network sizes.

Fig. 4.12 shows a comparison between four routing algorithms when applied to this larger networks. We note again that the DRA we propose presents the best performance while the LTTR is unable to guarantee acceptable delays. This confirms the viability and effectiveness of the DRA when increasing network size.



(a) 32x32 Lattice

(b) Small-world (32x32)

Figure 4.12: Change in $\bar{\omega}(\lambda)$ (mean delay, measured in minutes) when varying λ (car generation rate, measure in vehicles per second) in large networks with traffic lights present, when routing using DUA (orange line), LTTR (dark grey line), the shortest path (light grey line), and the DRA (blue line - using the best performing value of α for each network and chosen using Fig. 4.11). $\hat{\omega}$ (the acceptable delay threshold) is shown by the dashed red line for each network.

4.7 The Effect of Penetration Rate on Performance

Finally, we investigate how performance and resilience depend on the fraction of cars in the network adopting DRA as opposed to shortest path routing. This is to assess whether a beneficial effect can be gained even when only a fraction of the cars on the road adopt the DRA as a routing algorithm.

We varied the percentage of vehicles using the DRA, whilst the remaining vehicles used the shortest path.

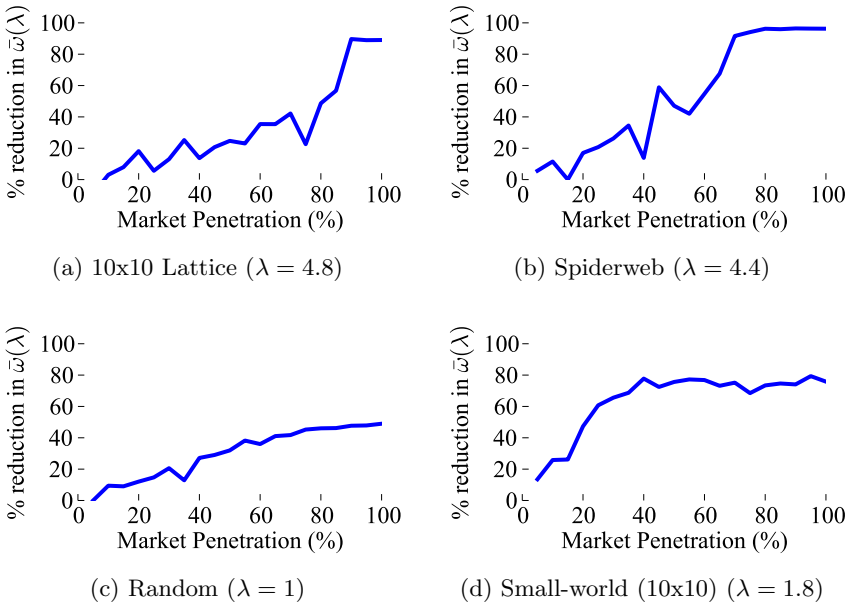
In the lattice network (Fig. 4.13a) we find that there is a gradual reduction in the mean delay as the percentage of vehicles using the DRA is increased. When the percentage of vehicles reaches 90% there is a sharp reduction in delays, and performance at 90% penetration rate is equal to performance at 100%.

Similarly in the spiderweb network (Fig. 4.13b) we observe the same improvement as the percentage of vehicles using the DRA is increased. We find that at low penetration rates there is an almost linear one-to-one relationship between the percentage reduction in $\bar{\omega}(\lambda)$ and the percentage of vehicles using the DRA. We observe that past 60% market penetration the effect of the DRA increases, and at 70% there is almost a 100% reduction in delays.

In the random network (Fig. 4.13c) we note that the effect of the DRA is less dramatic. Increasing the market penetration rate still has an approximately linear relationship to the reduction in $\bar{\omega}(\lambda)$, however, the relationship is of the order two-to-one, meaning that when the market penetration rate is at 80% we expect to see a 40% reduction in delays, and so on.

In contrast to the other networks we find that the small-world network (Fig. 4.13d) we observe a logarithmic increasing relationship between the market penetration and reduction in delays, between values of market penetration between 5% and 60%, showing that even small market penetration rates can yield high reductions in delays, for example at 30% market penetration we observe a 70% reduction in delays.

In summary, we have found that a clear beneficial effect is present even when a fraction of the vehicles in the network adopt the DRA, and the remaining vehicles are routed using



(a) 10x10 Lattice ($\lambda = 4.8$)

(b) Spiderweb ($\lambda = 4.4$)

(c) Random ($\lambda = 1$)

(d) Small-world (10x10) ($\lambda = 1.8$)

Figure 4.13: Percentage reduction in $\bar{\omega}(\lambda)$ (mean delay), in comparison to shortest path routing, when varying the percentage of vehicles using the DRA (vehicles not using the DRA are following the shortest path). Simulations were performed in networks with traffic lights, and the value of λ used varies in each network (as indicated in the sub-captions).

the shortest path. More notably, we observe again that the degree of improvement as a function of the adoption rate depends strongly on the network structure being considered.

4.8 Discussion

In this chapter, we presented a completely decentralised method to enable co-operative routing between vehicles.

The method demonstrates excellent results when compared with the shortest path, and even against an iterative DUA approach in some networks. The key feature of the proposed algorithm is that it relies on a local cost function where the relative weight between the shortest path to destination and road congestion is determined by a control parameter α. We convincingly showed that varying alpha has an effect on the performance of the DRA and can enhance the resilience of the road network that becomes able to cope with higher car generation rates for certain values of the control parameter. Simulations confirmed the ability of the DRA to guarantee better performance than other routing strategies including the iterative DUA where the routing problem is solved off-line. Remarkably, we showed the effect of changing the network structure and how the performance of the algorithm and choice of α are affected by such variations. We also showed that the observations made for smaller networks scale up to larger networks and that the full benefits of the DRA do not require all vehicles in the network to adopt the new routing strategy. This latter observation can be particularly relevant in the short/medium-term where automated vehicles will share the road network with human-driven ones. Indeed our findings show that better routing of the automated vehicles could have beneficial effects on the network as a whole with varying degrees of penetration rates being required depending on the

network structure.

In the next chapter, we will move to the problem of intersection control. We will explore our own method of decentralised intersection control based on the principle of work conservation.

Chapter 5

A Novel Approach to Intersection Control

Having studied a new decentralised strategy for routing vehicles that require communication between each vehicle and the infrastructure, we now move to the problem of improving traffic congestion by proposing a novel strategy for traffic light control.

When compared to the algorithms described earlier in Chapter 3 we notice the following key properties of our approach:

- The system we present is decentralised. That is, there is no traffic light controller that knows the state of all the other traffic lights. Rather, each traffic light controller only knows information it has measured or received from neighbouring intersections.
- The decentralised nature of the system makes it naturally *scalable*. In principle, since each controller only takes as input the state of its neighbours, it does not need to be reconfigured every time a new controller is inserted in the network.

Previous methods rely on turning ratios to determine the pressure of each phase, however, proliferation of Vehicle-to-Infrastructure (V2I) communication means that direct communication between vehicle and traffic light could soon be possible.

We propose a decentralised work conserving controller that communicates directly with vehicles and with neighbouring intersections. This information allows for the controller to know the exact incoming and outgoing lane of a vehicle. We propose two variants of a work conserving controller, one which detects the presence of congested roads, and one which estimates the exact capacity of neighbouring intersections and uses bounded queue lengths to determine the pressure.

We also tackle the problem of when the stage should be changed by introducing several stage duration algorithms. These algorithms calculate the correct green time for a stage based on the percentage of the current queue that should be cleared in the current stage (which for our purposes we set to 100%).

In the rest of this Chapter, after presenting the methodology for traffic intersection control,

- We formally prove that the algorithms are always able to move vehicles whenever the downstream intersections have available capacity. We relate this to the property of work-conservation.
- We discuss the complexity of the algorithms. Specifically, we show that workload for each intersection is independent of the number of intersections in the network and that it is bounded by physical constraints such as the number of neighbouring intersections.
- We validate our algorithms via microscopic traffic simulation. Specifically, we evaluate the behaviour of the algorithms in synthetic networks and under several traffic conditions.

5.1 The Proposed Architecture

We propose that a controller is present at each traffic light controlled intersection. As shown in the schematic architecture of Figure 5.1, controllers implement a closed loop system around the queues at the intersections. Controllers are also able to communicate with each other and with vehicles approaching the intersection. Essentially, the proposed system consists of the following components:

Sensors Manager responsible for collecting data at the traffic intersection and providing this information to the controller. Specific sensors that might be employed are inductance loops, microwave emitters, and cameras. The types of data collected from sensors include the queue length and vehicle speed.

Communications Manager responsible for (i) exchange of data between neighbouring intersections and (ii) exchange of data between controllers and approaching vehicles. In particular, neighbouring intersections share their available capacity and approaching vehicles communicate their intended route to the intersection, which combined with sensor data on the number of vehicles approaching allows the controller to determine accurate queue lengths and demand for neighbouring intersections.

Controller responsible for implementing the algorithms described in Section 5.3.

Traffic Lights responsible for displaying the red and green lights as an output of the controller. Traffic lights are essentially the *actuators* of the actions set by the controller.

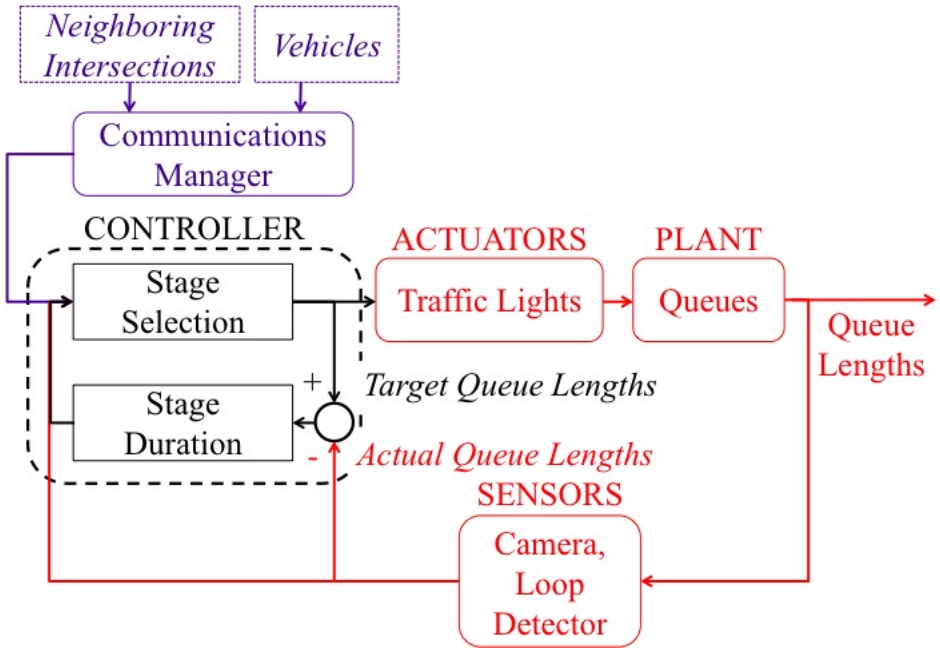


Figure 5.1: Schematic diagram illustrating the architecture of the decentralized controller presented in this chapter. The controller outputs the best stage choice (a combination of green lights at the intersection) as the control input to the traffic lights, in an attempt to control the queue sizes at the intersection. The controller also calculates a stage duration for each stage. Intersections communicate with sensors, with neighbouring intersections and also with approaching vehicles in order to gather data required to decide on the next state of the traffic lights. The communication manager and sensors also provide the input for the feedback loop used to adjust the stage duration.

We will assume that the necessary architecture is in place for the controller to implement the algorithms described and tested in this chapter. The controller consists of two algorithms which are responsible for determining, at the beginning of each stage: (i) the best combination queues to be released (i.e. the next stage) and (ii) the stage duration. The controller is responsible for triggering a new stage calculation, and this is implemented by means of a system clock.

5.2 Problem Formulation

Here we introduce the traffic model used throughout the chapter. For the convenience of the reader, we also provide a list of symbols in Appendix B.2. As we will not be revisiting the Decentralised Routing Algorithm (DRA) in the following chapters we will redefine any notation here as specific to the intersection control algorithms.

5.2.1 Modelling Networks of Intersections

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a directed graph, with $\{v_i\}_{i\in\mathcal{V}}$ being the set of vertices and $\{(v_a, v_b)\}_{(a,b)\in\mathcal{E}}$ be the set of edges [33]. We recall that the topology of a graph can be described by its

the adjacency matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$.

In the context of this chapter, the vertices of the graph will physically represent intersections, while the edges will physically represent roads. Edges are then directional as defined by the direction in which traffic flows along the road. Edges which flow into an intersection are input links to the intersection, whilst edges which flow out of an intersection are output links. We will say that two intersections (say v_a and v_b) are (a, b) *adjacent* if there is an output link of v_a which is an input link to v_b. We also refer to *adjacent* vertices as neighbours.

We consider the set of intersections denoted by $\mathcal{V}$. The i-th intersection is characterized by a set of input links $l \in I_i$ and output links $m \in O_i$. A phase, denoted by $j = \{l, m\}$, is any possible movement from an input link to an output link (this may also be referred to as the j-th phase or j-th queue). The number of possible phases at the i-th intersection is denoted as n_i, and the set of all phases can be expressed as $\sigma_i = \{j_1, ..., j_{n_i}\}$.

The number of vehicles (in a queue) associated with the j-th phase, at some discrete time step k, is denoted $x_i^j(k)$. We can refer to the total number of vehicles on input link l as $x_i^l(k)$, and on output link m as $x_i^m(k)$. Vehicles join the j-th phase at the the i-th intersection at the rate $\lambda_i^j(k)$, and can leave the j-th phase at the constant saturation rate μ_i^j. The status of the light controlling the j-th queue is given by $g^j(k) \in \{0, 1\}$, where 1 represents green (and 0 represents red).

In a network of intersections, we recall that output links associated with a given phase can become full and unable to take any more vehicles. We denote c_i^m as the capacity of the m-th output link associated to the phase $j = \{l, m\}$. We denote $\epsilon_j(x_i^m(k), c_i^m)$ as a function which gives 1 when there is available capacity on the output link of phase j, and 0 otherwise, such that,

$$\epsilon_j(x_i^m(k), c_i^m) = \begin{cases} 1 & \text{if } x_i^m(k) < c_i^m \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The service rate, say $s_i^j(k)$, for the j-th phase is the rate at which vehicles leave the j-th queue at time step k, taking into account whether or not it has received a green light, the saturation rate, and $\epsilon_j(x_i^m(k), c_i^m)$, such that,

$$s_i^j(k) = g_i^j(k) \cdot \epsilon_j(x_i^m(k), c_i^m) \cdot \mu_i^j \tag{5.2}$$

The traffic lights are controlled by selecting a *stage*, which is a set of compatible phases. We denote an arbitrary stage as $q = \{g_i^1, ..., g_i^{n_i}\}$, which in real terms defines what phases receive a red or green light (i.e. the value of $g_i^j(k)$ for all phases). We refer to an arbitrary stage as the q-th stage. The total number of stages is denoted as z_i, and the set of all possible stages as $\mathcal{P}_i$. In our framework the control input at time k is the stage being chosen, i.e. $\mathbf{u}_i(k) \in \mathcal{P}_i$ (note that $\mathbf{u}_i(k)$ is an $n_i \times 1$ column vector).

The state of the intersection is an $n_i \times 1$ column vector denoted $\mathbf{x}_i(k)$, which gives the length of the queues at the i-th intersection (i.e. $\mathbf{x}_i := [x_i^1, \ldots, x_i^{n_i}]^T$).

We denote by P_i the $z_i \times n_i$ stage matrix which is the stack of all stages in $\mathcal{P}_i$. The $n_i \times 1$ column vector $\mathbf{p}_i^q$ is the transpose of the q-th row of P_i, and hence is the column

vector representation of the q-th stage. We refer to the j-th row of the vector $\mathbf{p}_i^q$ as $p_i^{q,j}$, which is the value of g_i^j for the q-th stage.

We also let: (i) $\mathbf{a}_i(k) := [\lambda_i^1(k), \ldots, \lambda_i^{n_i}(k)]^T$ be the $n_i \times 1$ column vector of arrival rates; (ii) $D_i(k)$ be the $n_i \times n_i$ diagonal matrix having on its main diagonal the saturation rates μ_i^j; (iii) E_i be the $n_i \times n_i$ diagonal matrix having the element in the j-th row and j-th column equal to $\epsilon_j(x_i^m(k), c_i^m) \quad m \in j \quad \forall \quad j \in \sigma$ (i.e. the value of $\epsilon_j(x_i^m(k), c_i^m)$ for the output link of every phase at the i-th intersection), and (iv) $\mathbf{s}_i(k)$ be the $n_i \times 1$ column vector of service rates at time step k.

Then, the dynamics for the i-th intersection in the network can be written in compact form as,

$$\mathbf{x}_i(k+1) = [\mathbf{x}_i(k) + \mathbf{a}_i(k) - \mathbf{s}_i(k)]_+ \tag{5.3}$$

where,

$$\mathbf{s}_i(k) = E_i(k) \cdot (D_i \cdot \mathbf{u}_i(k)) \tag{5.4}$$

(Note: $[x]_+ \equiv \max\{x, 0\}$. For a generic n-dimensional vector, $[x]_+$ is a component-wise operation.)

For example, the state space model of a junction with two queues becomes,

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} - \begin{bmatrix} \epsilon_1 & 0 \\ 0 & \epsilon_2 \end{bmatrix} \cdot \left(\begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix} \cdot \begin{bmatrix} p_i^{q,1} \\ p_i^{q,2} \end{bmatrix} \right) \tag{5.5}$$

Measuring the Queue Length

In the model used by Varaiya [106], it is assumed that for every phase a vehicle joins a separate queue, and so there is no *head of line blocking*, whereby a vehicle belonging to a particular phase blocks the front of a queue, with vehicles behind being part of a different phase. This assumption is convenient, however, it is not always realistic, and we consider several different models which affect controller design. Here we provide the following descriptions to clarify how queue length may be measured, depending on which definition of the queue is chosen:

All-to-One Queue The queue length is determined by the number of vehicles occupying an edge, which may be thought of as a road. No matter what phase a vehicle belongs to it will be included in a total queue length for the edge. An intersection with 4 input roads will have 4 queues.

Some-to-One Queue The queue length is determined by the number of vehicles occupying an input link, which may be thought of as lanes within a road. No matter what phase a vehicle belongs to it will be included in a total queue length for the input link it occupies. An intersection with 4 input roads, each with 2 input lanes, will have a total of 8 input links and therefore 8 queues.

One-to-One Queue The queue length is determined by the phase a vehicle belongs to. All vehicles in a queue will make the same movement at the intersection. An intersection with 12 phases will have 12 queues, no matter how many input links at the intersection.

It is conceivable that we may wish to measure the number of vehicles, the edge occupancy, or some other function to determine queue lengths. If the number of vehicles is used then this can be counted for every phase, but if occupancy is used or measured by a sensor, then it represents the queue on an edge (all-to-one) or in a lane (some-to-one).

In order to measure queue length in the one-to-one case, it must be clear to which phase a vehicle belongs, either due to the position of the vehicle or by V2I communication. The most common solution found in the literature is to use or estimate the turning ratios at the intersection, although this would only give an approximation of the current state [46,106].

These definitions provide the necessary components to design a controller which selects the best stage at each time-step. We further establish the notation for the ease of calculating the stage and stage duration. The state of the i-th intersection $\mathbf{x_i}$ is expressed in vector notation, as is the notation for the set of all possible stages $\mathcal{P}$.

5.3 Control Algorithm Design

5.3.1 Overview

In this section, we present the main logical steps for the algorithms of the controller in Figure 5.1. To the best of our knowledge the algorithms presented are entirely new to the literature, although draw on the principles of work-conservation mentioned in the literature review and studied by other researchers. In particular the concept of pressure propagation introduced in Section 5.5.1 and studying the effect of network topology using algorithms of this nature has not been investigated before.

We first present the *Stage Selection Algorithms*, which determines which stage will be unlocked (i.e. the value of $u_i(k)$). We present two possibilities to select the stage to be unlocked. The algorithms differ in the information that they receive as input from nearby controllers. Namely, we introduce:

The Congestion-Aware Stage Selection Algorithm, whereby controllers use queue lengths to determine the stage with the maximum benefit to unlock, but do not include any queues for which the output links are already full.

The Capacity-Aware Stage Selection Algorithm, whereby controllers calculate the available capacity at their output links and use these capacities to calculate the maximum available benefit of actuating a particular stage (i.e. the stage with the maximum number of vehicles is released, but only cars which are moving into an output link with enough capacity are included in this calculation)

On inspection, it can be seen that the capacity-aware algorithm is simply an enhanced version of the congestion-aware algorithm. Next, we present the stage duration algorithms,

which calculate the length of time between subsequent stage changes. In this case, we propose three different solutions to determine the duration of each stage:

The Tmin/Tmax Stage Duration Algorithm, whereby the number of vehicles serviced during a stage is measured, and the stage duration is lowered/raised towards a minimum/maximum value dependent on the number of vehicles serviced.

The Proportional Stage Duration Algorithm, whereby the number of vehicles serviced during a stage is measured, and the stage duration is adjusted using an estimation of the error in the previous stage duration.

The Model-Based Stage Duration Algorithm, whereby the duration is calculated so as to reduce the queue by a certain fraction of the current queue length, taking into account the rate at which vehicles leave and enter the queue during the stage.

5.3.2 Stage Selection Algorithm Design

The stage selection algorithm maximises a utility function, denoted J_i^q, which is dependent on the stage, q, being considered. That is, the control input is chosen in a way such that

$$\mathbf{u}_i = \max_{\mathbf{p}_i^q \in \mathcal{P}_i} J_i^q(\mathbf{p}_i^q). \tag{5.6}$$

where as defined above $\mathbf{p}_i^q$ is the vector of the q-th stage of the i-th intersection, synonymous with the transpose of the q-th row of the matrix of all stages, P_i.

Congestion-Aware Stage Selection

The concept behind the Congestion-Aware stage selection algorithm is to allow each intersection to select a stage with the following characteristics: (i) it releases the largest number of vehicles at the input links; (ii) it does not include vehicles with no available capacity downstream. This concept is implemented in Algorithm 1.

The algorithm maximises the number of vehicles that are given a green light in the selected stage, discounting any queues for which the output link has no available capacity.

The algorithm at the i-th intersection checks, for each queue, if there is available capacity downstream. The algorithm creates a copy, $\tilde{P}_i$, of the stage matrix, P_i, which removes any phase from the stage matrix which has no available capacity in the output link. Specifically, it computes,

$$\tilde{P}_i = P_i \cdot E_i(k) \tag{5.7}$$

Matrix $\tilde{P}_i$ is used to calculate the utility function, J_i, whose q-th element is the value of $J_i^q(\mathbf{p}_i^q)$ as,

$$J_i = \tilde{P}_i \cdot \mathbf{x}_i \tag{5.8}$$

By the construction of $\tilde{P}_i$, the queues at intersection i that do not have available capacity downstream are excluded in the computation of the utility function. The stage

that is returned by the algorithm is the one corresponding to the largest element of J_i.

Algorithm 1 Congestion-Aware Stage Selection Algorithm

Communication with vehicles to determine desired outgoing lane
$\mathbf{x}_i \leftarrow$ vehicle queues
Compute $E_i(k)$
$\tilde{P}_i \leftarrow P_i \cdot E_i$
$J_i \leftarrow \tilde{P}_i \cdot \mathbf{x}_i$
$q \leftarrow$ index corresponding to maximum element in the vector J_i
$\mathbf{u}_i(k) = \mathbf{p}_i^q$
return $\mathbf{u}_i(k)$

Capacity-Aware Algorithm

The capacity-aware algorithm presented here calculates at each time step the total number of vehicles that wish to use a downstream edge and compares that to the exact number of vehicles that the downstream edge is able to accommodate. Where multiple queues share a downstream edge during a stage, their queue lengths are then adjusted (if necessary), so that their sum is never greater than the available capacity of the downstream edge. This concept is implemented in Algorithm 2.

Essentially, the algorithm first computes an $n_i \times n_i$ matrix describing whether two queues are both open during a given stage and using the same downstream link. We call this matrix $\tilde{L}_i^q(p_i^q)$, and we describe its computation in algorithm 3. The sum of all elements in the j-th row of $\tilde{L}_i^q(p_i^q)$ is denoted by $\tilde{\iota}_i^{q,j}$, which is the total number of phases sharing the output link of j-th phase during the q-th stage.

Once $\tilde{L}_i^q$ is calculated, the total demand for each downstream link can be calculated by finding the dot product of $\tilde{L}_i^q$ and the vector of measured queue lengths $\mathbf{x}_i$. The output is a set of queue lengths which are based on the combined demand for the output links, which we denote $\tilde{\mathbf{x}}_i^q$ (i.e. the j-th element of $\tilde{x}_i^q$ is the total demand for the output link, m, of the j-th queue). We refer to the j-th element of $\tilde{\mathbf{x}}_i^q$ as $\tilde{x}_i^{q,j}$

Using the data on available capacity received from sensors and neighbouring intersections, the j-th element of $\tilde{\mathbf{x}}_i^q$ is either left equal to the total demand for the downstream link of the j-th queue or reduced to match the available capacity, whichever is smaller. Where multiple queues share an output link, the final demand must be split between them. The j-th element of $\tilde{x}_i^q$ is divided by the total number of queues using the downstream edge of the j-th phase during the q-th stage (i.e. $\tilde{\iota}_i^{q,j}$).

We define $\upsilon(\tilde{x}_i^{q,j}(k), c_i^m), \quad m \in j$ as a function relating queue length for the j-th phase and the capacity of its outgoing link m.

$$\upsilon(\tilde{x}_i^{q,j}(k), c_i^m) = \begin{cases} \frac{\tilde{x}_i^{q,j}(k)}{\tilde{\iota}_i^{q,j}} & \text{if } \tilde{x}_i^{q,j}(k) < (c_i^m - x_i^m(k)) \\ \frac{(c_i^m - x_i^m(k))}{\tilde{\iota}_i^{q,j}} & \text{otherwise} \end{cases} \tag{5.9}$$

where $c_i^m - x_i^m(k)$ is a computation of the available number of vehicle spaces in the output link m.

We then denote as $\Upsilon_i^q(\tilde{\mathbf{x}}_i^q)$ the $n_i \times 1$ column vector, obtained by stacking the functions $\upsilon(\tilde{x}_i^{q,j}(k), c_i^m)$ for each of the n_i phases.

The utility function in (5.6) is finally calculated as,

$$J_i^q(\mathbf{p}_i^q) = \mathbf{p}_i^{qT} \cdot \Upsilon_i^q(\tilde{\mathbf{x}}_i^q) \tag{5.10}$$

Algorithm 2 Capacity-Aware Stage Selection Algorithm

Communication with vehicles to determine desired outgoing lane
$\mathbf{x}_i \leftarrow$ vehicle queues
for $\mathbf{p}_i^q \in \mathcal{P}_i$ **do**
  Implement algorithm 3
  $\tilde{L}_i^q(\mathbf{p}_i^q) \leftarrow \text{GET}\tilde{L}(\mathbf{p}_i^q)$
  $\tilde{\mathbf{x}}_i^q(\mathbf{p}_i^q) \leftarrow \tilde{L}_i^q \cdot \mathbf{x}_i$
  for phase $j \in \sigma_i$ **do**
    Communication with neighbouring intersections
    $c_i^m \leftarrow$ maximum capacity of outgoing lane of j-th queue
    $\tilde{\iota}_i^{q,j} \leftarrow$ sum of j-th row of $\tilde{L}_i^q(\mathbf{p}_i^q)$
    j-th element of $\Upsilon_i^q \leftarrow \upsilon(\tilde{x}_i^{q,j}(k), c_i^m)$
  $J_i^q(\mathbf{p}_i^q) = \mathbf{p}_i^{qT} \cdot \Upsilon_i^q(\tilde{\mathbf{x}}_i^q)$
$\mathbf{u}_i(k) = \max_{\mathbf{p}_i^q \in \mathcal{P}_i} J_i^q(\mathbf{p}_i^q)$
return $\mathbf{u}_i(k)$

Algorithm 3 Construct $\tilde{L}$ Matrix from Stage

procedure $\text{GET}\tilde{L}(p_i^q)$
  $\tilde{L} \leftarrow n_i \times n_i$ matrix of all 0
  for queue $a = 1, \ldots, n_i$ **do**
    for queue $b = 1, \ldots, n_i$ **do**
      if a **and** b receive a green light under p_i^q **then**
        if a **and** b use the same output link **then**
          $\tilde{L}(a, b) \leftarrow 1$
  return $\tilde{L}$

5.3.3 Stage Duration Algorithm

The stage selection algorithm is responsible for setting a control signal, $u_i(k)$, in order to unlock one of the stages from the set of feasible stages $\mathcal{P}_i$. The stage duration algorithm is responsible for setting the duration of the control signal. We now present three alternative choices to determine the duration of $u_i(k)$.

At the start of each stage, the total number of vehicles the algorithm wishes to remove from the i-th intersection is calculated. This value is a fraction of the total number of vehicles in the queues released during the stage. We term the number of vehicles to be removed from the i-th intersection during stage q as $\delta_i^q(k)$, such that,

$$\delta_i^q(k) = \bar{\eta}_i \cdot p_i^q \cdot x_i(k), \tag{5.11}$$

where $\bar{\eta}_i$ (a design parameter) is the target fraction of vehicles to remove from the open queues during a single stage and is a constant.

Let $\tau_i^q(k)$ be the duration of the q-th stage computed by the controller at time step k. We will then denote by $\gamma_i^q(k, k + \tau_i^q(k))$ the number of vehicles that departed during the q-th stage (i.e. between time step k and time step $k + \tau_i^q(k)$, obtained by measuring the number of vehicles that cross the stop line at the traffic light, or by communication with the vehicles.

At the beginning of the q-th stage, a system clock is set to $\tau_i^q(k)$. The clock is decremented at each time step, and when the system clock reaches 0 a new stage is selected.

The algorithms we propose adjust the stage duration at the end of each stage, according to the difference between $\delta_i^q(k)$ and $\gamma_i(k, k + \tau_i^q(k))$.

The first two algorithms we propose (Tmin/Tmax and Proportional) adjust the stage duration at the end of each stage, according to the difference between $\delta_i^q(k)$ and $\gamma_i(k, k + \tau_i^q(k))$. The third algorithm (Model Based) sets the stage duration at the beginning of the stage.

Tmin/Tmax Stage Duration Algorithm

When implementing the Tmin/Tmax stage duration algorithm a minimum and maximum value is set for the duration of any given stage, which we denote $\bar{\tau}_{\min}$ and $\bar{\tau}_{\max}$ respectively. The initial duration of any stage is the mean of this minimum and maximum value.

When a stage ends, its duration is adapted to reflect its performance. If the same stage is selected on subsequent cycles then it will use this new adapted duration value. Performance is measured against the desired number of vehicles that were to be cleared from the intersection, δ_i^q. If too few vehicles are cleared, then the stage duration is moved closer to the value of $\bar{\tau}_{\max}$, if more vehicles were cleared then the stage duration will be moved closer to the value of $\bar{\tau}_{\min}$. Further explanation can be found in algorithm 4.

Algorithm 4 Tmin/Tmax Stage Duration Algorithm

Compute stage duration, τ_i^q , for stage $\mathbf{p}_i^q$
$\mathbf{p}_i^q \leftarrow$ output from stage selection algorithm
if no green time previously calculated for stage $\mathbf{p}_i^q$ **then**
  $\tau_i^q \leftarrow \frac{\bar{\tau}_{\max}+\bar{\tau}_{\min}}{2}$
System Clock $\leftarrow \tau_i^q$
Target number of vehicles to service
$\delta_i^q(k) \leftarrow \bar{\eta}_i \cdot \mathbf{p}_i^q \cdot \mathbf{x}_i(k)$
while System Clock > 0 **do**
  Wait until the System Clock reaches 0
$\gamma_i^q \leftarrow$ total vehicles departed during the stage
if $\gamma_i^q > \delta_i^q$ **then**
  $\tau_i^q \leftarrow \frac{\bar{\tau}_i^q+\bar{\tau}_{\min}}{2}$
else if $\gamma_i^q < \delta_i^q$ **then**
  $\tau_i^q \leftarrow \frac{\tau_i^q+\bar{\tau}_{\max}}{2}$

Proportional Stage Duration Algorithm

The proportional stage duration algorithm is similar to the Tmin/Tmax algorithm. The stage duration is adjusted at the end of a stage based on the number of vehicles that were cleared from the intersection. In order to make the adjustment, the error in the stage duration is estimated and a proportional control algorithm is applied. The initial stage duration for all stages is set to some starting value, denoted as τ_0.

The stage duration of the q-th stage is recalculated according to an estimated error, β_i^q, in the previously calculated stage duration. This error is calculated such that,

$$\beta_i^q = \frac{\delta_i^q - \gamma_i^q}{\delta_i^q} \cdot \tau_i^q \tag{5.12}$$

On inspection, we see that this error is an estimate of the fraction of the time that should be added or removed in order that the values of δ_i^q and γ_i^q are equal.

We apply a proportional control law, with gain K_P, such that,

$$\tau_i^q = \tau_i^q + K_P \cdot \beta_i^q \tag{5.13}$$

This stage duration algorithm gradually adjusts to changing traffic conditions. Further explanation can be found in algorithm 5.

Model Based Stage Duration Algorithm

The model based stage duration algorithm is proposed based on the model shown in (5.3).

The model based algorithm calculates the stage duration required to clear the queues by the given proportion, $\bar{\eta}_i$, taking into account the arrival rate $\mathbf{a}_i$ and departure rates D_i of vehicles. For a single intersection of arrival rate λ, departure rate μ, and current queue

Algorithm 5 Proportional Stage Duration Algorithm

Compute stage duration, τ_i^q , for stage $\mathbf{p}_i^q$
Traffic light enters state $\mathbf{p}_i^q$
if no green time previously calculated for stage $\mathbf{p}_i^q$ **then**
    Assign predetermined value
    $\tau_i^q \leftarrow \tau_0$
System Clock $\leftarrow \tau_i^q$
Target number of vehicles to service
$\delta_i^q(k) \leftarrow \bar{\eta}_i \cdot \mathbf{p}_i^q(k) \cdot \mathbf{x}_i(k)$
while System Clock > 0 **do**
    Wait until the System Clock reaches 0
$\gamma_i^q \leftarrow$ total vehicles departed during the stage
Estimated error in current green time
$\beta_i \leftarrow \frac{\delta_i^q - \gamma_i^q}{\delta_i^q} \cdot \tau_i^q$
$\tau_i^q \leftarrow \tau_i^q + K_P \cdot \beta_i$

length x, then the period of time, τ, required to clear the required number of vehicles, $\bar{\eta} \cdot x$, is given by,

$$\tau = \frac{\bar{\eta} \cdot x}{\mu - \lambda} \tag{5.14}$$

where $\mu - \lambda$ is the real rate at which the queue length is reduced, and $\bar{\eta} \cdot x$ is the amount the queue needs to be reduced by, hence the result it the time required to do this. As $\mu - \lambda \to 0$ the time required increases and vice-versa. In this model, it is assumed that $\lambda \leq \mu$, and can, therefore, we cannot have a negative value of τ. In practice, this assumption is reasonable, as vehicles cannot join a queue faster than space is being made available.

For the i-th intersection using stage $\mathbf{p}_i^q$, we find that the time required can be given by,

$$\tau_i^q(k) = \frac{\delta_i^q(k)}{\mathbf{p}_i^q \cdot ((D_i(k) \cdot \mathbf{p}_i^{qT}) - \mathbf{a}_i(k))} \tag{5.15}$$

Note that we do not account for congested downstream links in this algorithm, hence the use of the saturation rates in D_i over the service rates $\mathbf{s}_i$.

Moving averages are computed for the values of μ_i^j and λ_i^j over a chosen time window. In contrast to the previous methods, the stage duration is calculated at the beginning of a stage. An overview of this method can be found in Algorithm 6.

Algorithm 6 Model Based Stage Duration Algorithm

Compute stage duration, τ_i^q , for stage $\mathbf{p}_i^q$

Traffic light enters state p^q

Target number of vehicles to service

$\delta_i^q(k) \leftarrow \bar{\eta}_i \cdot p_i^q(k) \cdot x_i(k)$

$D_i \leftarrow$ mean departure rates for queues

$\mathbf{a}_i \leftarrow$ mean arrival rates for queues

$\tau_i^q \leftarrow \frac{\delta_i^q(k)}{\mathbf{p}_i^q \cdot ((D_i(k) \cdot \mathbf{p}_i^{qT}) - \mathbf{a}_i(k))}$

5.4 Properties of the Controller

5.4.1 Work Conservation

To better characterise the controller, we introduce a definition of *efficiency*, or *work conservation* for a policy, u_i. This notion is formalized with the following definition.

Definition 1. *Let u_i be a policy for intersection i and let $x_i^j > 0$ for some $j = 1, \ldots, n_i$. Assume that some capacity is available to the nearby intersections. The policy is said to conserve work for intersection i if the only stages that can be unlocked are those with available capacities downstream.*

The definition above implies that a sufficient condition for the policy to move vehicles is to have available capacity downstream. Our interest in work conservation is then motivated by the fact that, intuitively, a loss of work for a policy is an indicator of inefficiency in the road network.

Our main result on work conservation can be stated as follows.

Theorem 1. *Both the congestion-aware and the capacity aware controllers are work preserving.*

Proof. We prove this by contradiction, by assuming that the congestion-aware and the capacity-aware controllers are not work preserving. More precisely, we assume that the policy set by the controllers can unlock a stage, say p_i^q with no available capacity downstream, even if other stages have available capacity.

Congestion-aware controller. Since, by contradiction, this controller is not work preserving, this means that, for at least one intersection in the network, say intersection i, the corresponding policy, u_i set by means of Algorithm 1 or 2 unlocks a service stage, say p_i^q, that has no available capacity downstream, even if other queues have available capacities. That is, p_i^q is characterized by the fact that:

- $x_i^j > 0$ for at least some of the queues unlocked during stage p_i^q;
- the capacities downstream to x_i^j's are equal to 0.

Recall the following steps from Algorithm 1

- compute $\tilde{P}_i x_i$;

- pick q as the index corresponding to the biggest element of the above vector.

Thus, since p_i^q is chosen by Algorithm 1, it means that q is the index corresponding to the largest element of $\tilde{P}_i x_i$. However, since there is no capacity downstream, then, by construction, the q-th element of $\tilde{P}_i x_i$ is equal to 0. Therefore, it follows that all the other elements of $\tilde{P}_i x_i$ are all equal to 0. This, however, can happen only if the elements of matrix $\tilde{P}_i$ are equal to 0. Now, all the elements of this matrix are set to 0 by the algorithm only if all the queues downstream the intersection are at their maximum capacity, i.e. if the queues downstream intersection i have no available capacity. This is, however, a contradiction as our argument was based on the fact that the controller picked stage p_i^q even if other stages had available capacity.

Capacity-aware controller. The proof for the capacity-aware controller follows exactly the same steps as the proof for the congestion-aware controller. It is therefore omitted here for the sake of brevity. □

5.4.2 Complexity of the algorithms

We remark here that the complexity of implementing these algorithms network-wide is $\mathcal{O}(1)$ in terms of the number of intersections in the network. The complexity of the algorithms computed at each intersection is dependent on the number of queues managed by that intersection, as well as the number of possible stages to be compared.

In regards to the stage selection algorithms:

- The congestion-aware stage selection algorithm is a sequence of $\mathcal{O}(1)$ and $\mathcal{O}(n)$ operations, where n is the number of queues at the i-th intersection, and therefore has overall $\mathcal{O}(n)$ complexity.

- The capacity-aware stage selection algorithm is $\mathcal{O}(m \times n)$ complex, where m is the number of possible phases z_i and n is the number of queues at the i-th intersection, n_i. The computation of $\tilde{L}$ is $\mathcal{O}(n_i^2)$, however, it only ever needs to be calculated once for each stage if it is stored in memory.

The stage duration algorithms:

- The Tmin/Tmax and proportional stage duration algorithm are $\mathcal{O}(n)$ complex, where n is the number of queues at the i-th intersection, n_i. Specifically it is the calculation of δ_i^q which is $\mathcal{O}(n)$ complex.

- The model based algorithm is $\mathcal{O}(n^2)$ complex, specifically the calculation of $D_i - A_i$, where n is the number of queues at the i-th intersection, n_i.

The number of queues at an intersection is physically limited. Take for example a 4-way intersection with 3 lanes per road direction (allowing 3 'straight-on' lanes and 1 right and left turn lane respectively). This would results in 36 queues ($n_i = 36$) and could operate using 4 possible stages ($z_i = 4$). The computational complexity is therefore low.

5.5 Developing an Advanced Algorithm to Address 'Head-of-line-blocking'

In Chapter 6 section 6.2, we show results for numerical validation of the above algorithms. We find that our intersection controller performs well in the lattice and random networks, but is disappointing in the small-world network (for further details, refer to Figures 6.3, 6.4, and 6.6). Observations of behaviour in the small-world network highlighted two possible causes,

Head-of-Line Blocking is a problem caused when the first vehicle in a queue cannot enter the next lane and blocks cars behind it which would otherwise be able to move on to the next intersection.

Short Road Blocking was a problem where busy short roads were unable to release their traffic due to sharing a junction with a busy long road. If another intersection needed the short road as an output link, it was never able to clear traffic, despite there being a high demand in the network for that link.

Short road blocking has been addressed in the literature by using measures such as road occupancy, rather than the number of vehicles, to calculate queue sizes. Road occupancy is normalised and therefore a short road which is full will have the same queue size as a long road which is full.

Here we propose a solution, termed the ***Pressure Propagating Controller (PPC)***, which addresses both short road blocking and head of line blocking. Our solution requires intersections to propagate queue lengths to downstream intersections. The advantages of our solution are:

- Shorter queues may extend their queues to account for upstream traffic at another intersection.
- Longer queues are still prioritised if a short road is not blocking upstream traffic, but the increase in pressure from an additional vehicle decreases the longer the queue is.
- Intersections downstream prioritise clearing links which are needed by vehicles causing head-of-line blocking.

5.5.1 The Pressure Propagating Controller

The PPC uses the same architecture found in Section 5.3. The key additions to the algorithm are:

- A vehicle's weight in a queue is inverse to its position in a lane (i.e. the further away from the stop-line it is, the less it increases the pressure of the queue it has joined).
- Intersections propagate the total number of vehicles waiting in a lane to the next intersection. The direction of propagation is in the direction that the first vehicle wishes to go (i.e. information is passed between neighbouring intersections on the number of vehicles being blocked).

- Information is propagated downstream by each intersection in the same manner, so that propagated pressures are cumulative. The direction of propagation from each input link is dictated by the intersection and done in the direction of travel of the first vehicle in an input links queue.
- Propagation ends when an output link is empty, or a cycle is formed (the input link of the original intersection is an output link of a downstream intersection).

The PPC uses a combination of queues at the intersection and the queues propagated from upstream intersections to pick the stage for the intersection. The result is that the stage selection algorithm:

- Releases the most potential pressure on the network due to head of line blocking.
- Conserves work by avoiding the servicing of empty queues or queues which have nowhere to go (due to congestion).
- Puts high pressures on downstream links which are causing traffic blockages further upstream.

The PPC communicates with neighbouring intersections. At every intersection, and for every lane, the intersection checks for any non-empty queues, and notes the turning movement of the vehicle at the front of the lane. The turning movement informs the current intersection what the next intersection (and lane) of that vehicle will be. Intersections pass both their own queue length to that lane, as well as any queue lengths received from previous intersections (see Figure 5.2). We refer to these cumulative queue lengths being passed to downstream intersections as the *pressure propagation*.

There are two conditions which halt pressure propagation:

- If the first vehicle in an input link wants to turn into an empty lane.
- If the propagating pressure from input link l forms a cycle and l becomes an output link for another intersection.

5.5.2 The Pressure Propagating Controller Stage Selection Algorithm

The purpose of this algorithm is to prioritise vehicles which are at the front of a queue, and which may, therefore, be blocking vehicles behind. Less priority is given to vehicles if they are far back in a queue, as it is assumed that they may not be able to move into their desired output link either due to a vehicle ahead of them or due to the output link having become congested. Intersections also prioritise the first vehicle in each input link by taking its route and asking the next intersection downstream to prioritise the stage which will make space available for that vehicle. Here we provide an overview of the algorithm, and further details will be presented elsewhere.

The PPC Stage Selection Algorithm is based on the congestion-aware algorithm of Section 5.3.2, however, it differs in three main ways from the cap-aware algorithm.

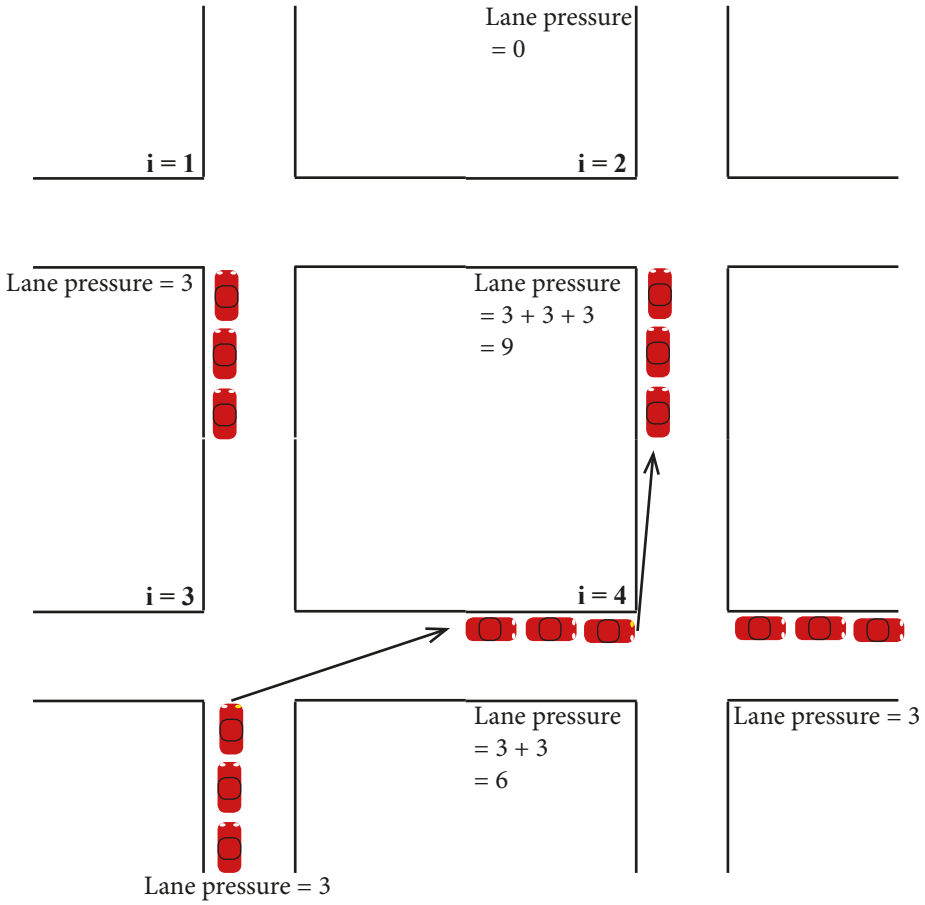


Figure 5.2: The pressure in each lane (in this case the queue length) is propagated towards the next lane in the direction of the first vehicle in the queue. Propagation is continuous until an empty lane is reached (which can be seen for $i = 2$), or the pressure re-enters the original lane, which would create a cycle.

- A vehicles contribution to queue length is inversely proportional to its position on an input link
- For the purpose of stage selection, congestion is detected using equation (5.1) and real-time capacity is not considered
- Actual queue lengths are propagated downstream in the direction of travel the first vehicle on an input link

More specifically, let $\mathcal{B}_i^l$ be the set of all vehicles on input link l at the i-th intersection, and $\mathcal{B}_i^j$ be the set of all vehicles assigned to phase j at the i-th intersection (s.t. $\mathcal{B}_i^j \subset \mathcal{B}_i^l \iff l \in j$).

We denote by $b = \{l, m\}$ the b-th vehicle in a queue, which has an input link, l, and an output link, m. We denote by $y_l^b(k)$ the position of the b-th vehicle in the queue on input link l (i.e. $y_l^b = 1$ for the vehicle at the front of input link l, $y_l^b = 2$ for the second vehicle in the queue, etc.).

We denote the position ranked queue length for the j-th phase as $\bar{x}_i^j$, and calculate it such that,

$$\bar{x}_i^j = \sum_{b \in \mathcal{B}_i^j} \frac{1}{y_l^b}, \quad l \in j \tag{5.16}$$

The vehicle at the front of the queue on input link l is denoted as $b_l^*(k)$ (i.e. $y_l^{b_l^*} = 1$). The PPC stage selection algorithm combines position ranked queue lengths with propagated pressures from upstream intersections. The sum of pressures propagated from upstream intersections to input link l is denoted as $\bar{r}_i^l(k)$. The combined queue length for the j-th phase, used to find the optimal stage at each intersection is denoted as $\hat{x}_i^j(k) = \bar{x}_i^j(k) + \bar{r}_i^l(k), \quad l \in j$. We refer to the $n_i \times 1$ column vector which is the stack of all $\hat{x}_i^j(k)$ as $\hat{\mathbf{x}}_i(k)$.

The pressure generated by input link l at time step k is the number of vehicles on input link l at time step k. We denote the pressure to be propagated as $r_i^{l,m}(k) = \bar{r}_i^l(k) + x_i^l(k), \quad m \in b_l^*(k)$. Pressure is always propagated in the direction of travel of the vehicle at the front of the queue on input link l.

The utility function for the q-th stage is calculated as $J_i^q(\mathbf{p}_i^q) = \hat{\mathbf{x}}_i^T(k) \cdot E_i(k) \cdot \mathbf{p}_i^q$.

Algorithm 7 provides further explanation and also clarifies the logic which halts the propagation of pressure along the network.

5.6 Discussion

In this chapter, we presented several decentralised algorithms for intersection control which rely on V2I communication to allow intersections to pick the best stage and stage duration at a given time step. These algorithms are intended to maximise the flow of vehicles through the network, and minimise delays to vehicles in the network. We have created 2 stage selection algorithms based on the principle of work-conservation (with accompanying

Algorithm 7 Pressure Propogation Stage Selection Algorithm

For every intersection
for $i \in \mathcal{V}$ **do**
  For every phase associated with the intersection
  for j in σ_i **do**
    Calculate the position ranked queue length
    $\bar{x}_i^j \leftarrow \sum_{b \in \mathcal{B}_i^j} \frac{1}{y_l^b}, \quad l \in j$
    Retrieve upstream pressures for the input link of the j-th phase
    $\bar{r}_i^l(k) \leftarrow$ pressure from upstream links into input link l
    if $\bar{r}_i^l(k)$ contains pressures propagated from input link l (i.e. a cycle has formed) **then**
      break
    else
      Calculate the overall pressure for the phase
      $\hat{x}_i^j(k) = \bar{x}_i^j(k) + \bar{r}_i^l(k), \quad l \in j$
      Get the output link of the vehicle at the front of input link l
      if input link l has an empty queue **then**
        $r_i^{l,m}(k) = 0$
      else
        $b_l^*(k) \leftarrow$ first vehicle in queue on input link l
        $r_i^{l,m}(k) = \bar{r}_i^l(k) + x_i^l(k), \quad m \in b_l^*(k)$
For every intersection
for $i \in \mathcal{V}$ **do**
  $E_i(k)$ is computed by the intersections
  $J_i^q(\mathbf{p}_i^q) = \hat{\mathbf{x}}_i^T(k) \cdot E_i(k) \cdot \mathbf{p}_i^q$
  return $J_i^q(\mathbf{p}_i^q)$

stage duration algorithms) and presented another algorithm which attempts to counter some of the problems observed in the simulations which now follow.

In Chapter 6 we will present results from simulations in synthetic networks, where we have tested each of these algorithms in combination. In Chapter 7 we show preliminary results for our algorithms from simulations of the city of Luxembourg.

Chapter 6

Numerical Validation of our Intersection Control Algorithms in Synthetic Networks

Here we present results from numerical simulations of the intersection control algorithms introduced in the previous Chapter. The results are divided over two sections:

In Section 6.2 we compare fixed cycle scenarios, with a mode stage duration of 31 seconds or 15 seconds, with the congestion-aware (algorithm 1) and capacity-aware (algorithm 2) stage selection algorithms in combination with Tmin/Tmax (algorithm 4), proportional (algorithm 5), and model-based (algorithm 6) stage duration calculation. A variety of parameters were tested, varying the maximum stage durations and starting stage durations in the controlled schemes. The resulting figures show the comparison of several scenarios, which are detailed in Table 6.2. These scenarios allow for the easy comparison of various combinations of stage selection and stage duration algorithm.

In Section 6.3 we compare the Pressure Propagating Controller (PPC) with two algorithms from the literature. Furthermore, we contrast these results with our findings from the congestion-aware and capacity-aware algorithms to draw conclusions about which intersection control strategy provides the best performance.

6.1 Simulation Methodology

Simulations were performed using Simulation of Urban Mobility (SUMO), which is an open source software platform suitable for microscopic traffic simulation [12]. Extensions were developed in Python which enabled the testing of our stage selection and stage duration algorithms. The intersection controller connects to the simulation through the SUMO API TraCI, and can request data such as the queue lengths, and also set the traffic lights in the simulation (see Figure 6.1). The system clock for each intersection is handled externally to SUMO, and will cause a stage change when it reaches 0.

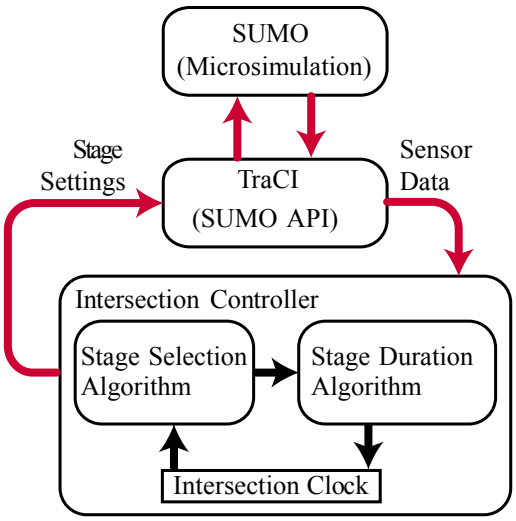


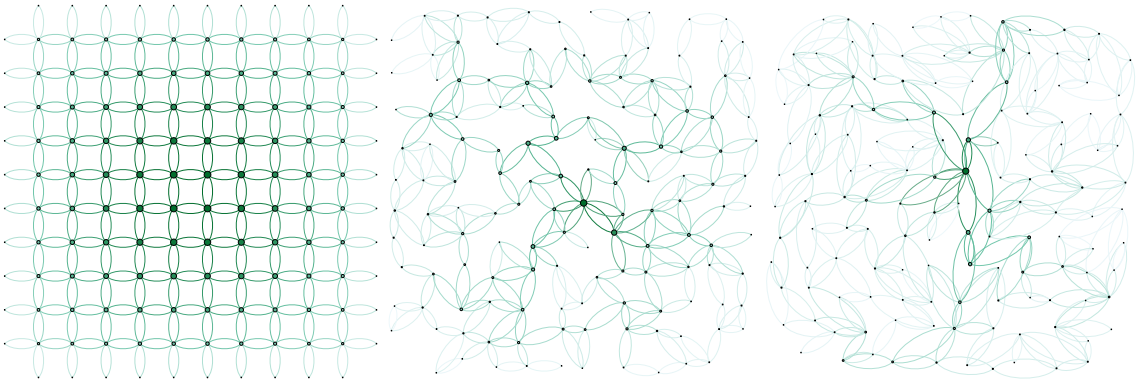
Figure 6.1: Simulation architecture demonstrating how the traffic microsimulation (written in C), connected to our intersection controller with its stage selection and stage duration algorithms via the SUMO API (TraCI).

We tested the controllers across 3 synthetic network topologies: a lattice network, a random network, and a small-world network. The random and small-world networks were created by rewiring the lattice network (following the same methodology mentioned in Chapter 4). This enabled us to keep the number of nodes and edges constant in each network, whilst varying the topology.

Table 6.1 shows the number of nodes, edges and traffic light controlled intersections in each network. Rewiring of the lattice network results in fewer or greater number of traffic lights required in the network (nodes that go from 3 or more undirected edges to 2 or fewer undirected edges no longer require traffic lights, and vice versa).

Figure 6.2 shows an analysis of the betweenness centrality of nodes in the networks. In the lattice network (Figure 6.2a). we find a high number of nodes with a high betweenness centrality, offering many shortest paths throughout the network, especially for cars moving through the centre of it. In the random network (Figure 6.2b) we observe fewer nodes with a high betweenness centrality, meaning there are fewer short routes between each node and more points where bottlenecks might occur. In the small-world network (Figure 6.2c) there is a clear central node with a high betweenness centrality, whilst nodes on the periphery have a very small betweenness centrality. This can be seen as a way of going from homogeneously loaded to heterogeneously loaded

In each simulation, cars were generated for a period of 1 hour at a constant rate ranging from 0.5 to 5 vehicles per second. The trips were generated at randomly between two nodes with a minimum distance of 900m for any given trip, and routes were calculated according to Dijkstra's algorithm [37]. The simulation was ended after 2 hours even if some vehicles had not completed their journeys. In order to avoid severely delayed vehicles from failing to register their trip duration (if they had not completed their journey when the simulation ended) any active vehicles in the network at the end of the simulation were



(a) Lattice (b) Random (c) Small-world

Figure 6.2: Comparison of betweeness centrality in the networks. Larger and darker nodes have a higher betweeness centrality in the network.

forced to complete their journey, and the time they had spent in the network was taken as the trip duration. For each simulation, a total of 5 simulations were run at every car generation rate with a different set of trips, in order to approximate any error in our results.

Statistics were recorded for the number of vehicles in the network, and the individual trip time and wait time of each vehicle.

Table 6.1: Network Parameters

Topology	**Nodes**	**Edges**	**Traffic Lights**
Lattice	140	440	100
Random	140	440	101
Small-world	140	440	92

6.1.1 Method for Estimating the Capacity of Roads

The purpose of a work-conserving intersection controller is to avoid servicing empty roads. However, in the presence of spill-over or gridlock in the outgoing lanes, a work conserving intersection controller can still be susceptible to a loss of work if green lights are given to vehicles with nowhere to go. Here we develop the concept of outgoing lane capacity and its computation.

The capacity is an estimate of the number of vehicles which can be accommodated by an outgoing lane. The estimate is based upon the length of the road, the length of a vehicle, and the average gap between vehicles when stopped. There are numerous possible methods for calculating the value, for example:

- Taking the position of the last vehicle in the queue, and calculating using the re-

maining length of road behind that vehicle.

- Assigning a nominal capacity for each road and a nominal capacity-requirement per vehicle.
- Estimating capacity as a function of road density or occupancy.

However, the key requirements for our estimation were that:

- The estimation should never be above 0 when the road is full (i.e. it should be conservative)
- The estimation should be feasible through estimated values that might be picked up by sensors in the road, or by cameras.

In order to fulfil these requirements, it was decided to use a simple estimation of road length, divided by the mean vehicle length and a constant vehicle gap. The vehicle gap could then be adjusted to ensure conservative estimates of capacity, and ensure good operation of the intersection controller.

$$c_i^j = \left\lfloor \frac{\lfloor \text{Road Length} \rfloor}{\text{Mean Vehicle Length} + \text{Vehicle Gap}} \right\rfloor \tag{6.1}$$

6.1.2 Performance Metrics Used to Evaluate the Algorithms

We consider 3 metrics when assessing the performance of our queue unlocking controller and method of stage duration calculation:

- Mean network flow
- Mean trip duration
- Mean waiting time

We consider *network flow* as the measure of how much traffic can be routed through the network and is measured in vehicles per hour. The higher the network flow the higher the capacity of the network. It is calculated such that,

$$\text{Network Flow} = \frac{\text{Num. Completed Journeys}}{\text{Time Period (Hours)}} \tag{6.2}$$

Mean trip time is measured in minutes and is the average time elapsed between a vehicles departure and arrival. A reduced travel time represents a tangible benefit to an individual driver, and so we compare the mean travel time achieved with increasing numbers of vehicles wishing to use the network.

The *mean waiting time* is measured in minutes and is calculated as the average time for which a vehicle is stopped (involuntarily) during its journey. A high waiting time would likely be perceived as undesirable by the driver, even if it correlated to higher network flow and a reduced average travel time.

Table 6.2: Selection of Controllers and Parameters Compared in the Figures on Synthetic Networks

Scenario	**Uncontrolled**	**Controlled Parameters**						
	Parameters	***Stage Selection Algorithm***	***Stage Duration Algorithm***					
				Tmin/Tmax		Proportional		Model Based
	Fixed Cycle		$\bar{\eta}_i$	*Tmin*	*Tmax*	$\mathcal{K}_P$	T_0	*Tmin*	*Tmax*
A	31	Congestion-Aware	1	5	55	0.15	30	5	55
B	15	Congestion-Aware	1	5	25	0.15	15	5	25
C	31	Capacity-Aware	1	5	55	0.15	30	5	55
D	15	Capacity-Aware	1	5	25	0.15	15	5	25

We also consider the *critical car generation rate*, which is the rate at which the number of vehicles entering the network causes the network to become severely congested and enter a state of gridlock. We find that this critical car generation rate is significantly modified when comparing fixed-interval traffic lights and our intersection controller.

6.2 Numerical Analysis of the Congestion-Aware and Capacity-Aware Stage Selection Algorithms

In Figures 6.3, 6.4 and 6.6, we show the network flow, mean trip duration and mean wait time for simulations the lattice, random and small-world networks. These figures show the high-level performance of the fixed-interval and controlled scenarios, such as the effect on the critical car generation rate. In Figures 6.5 and 6.7 we show more detailed comparison of small variations in algorithm performance.

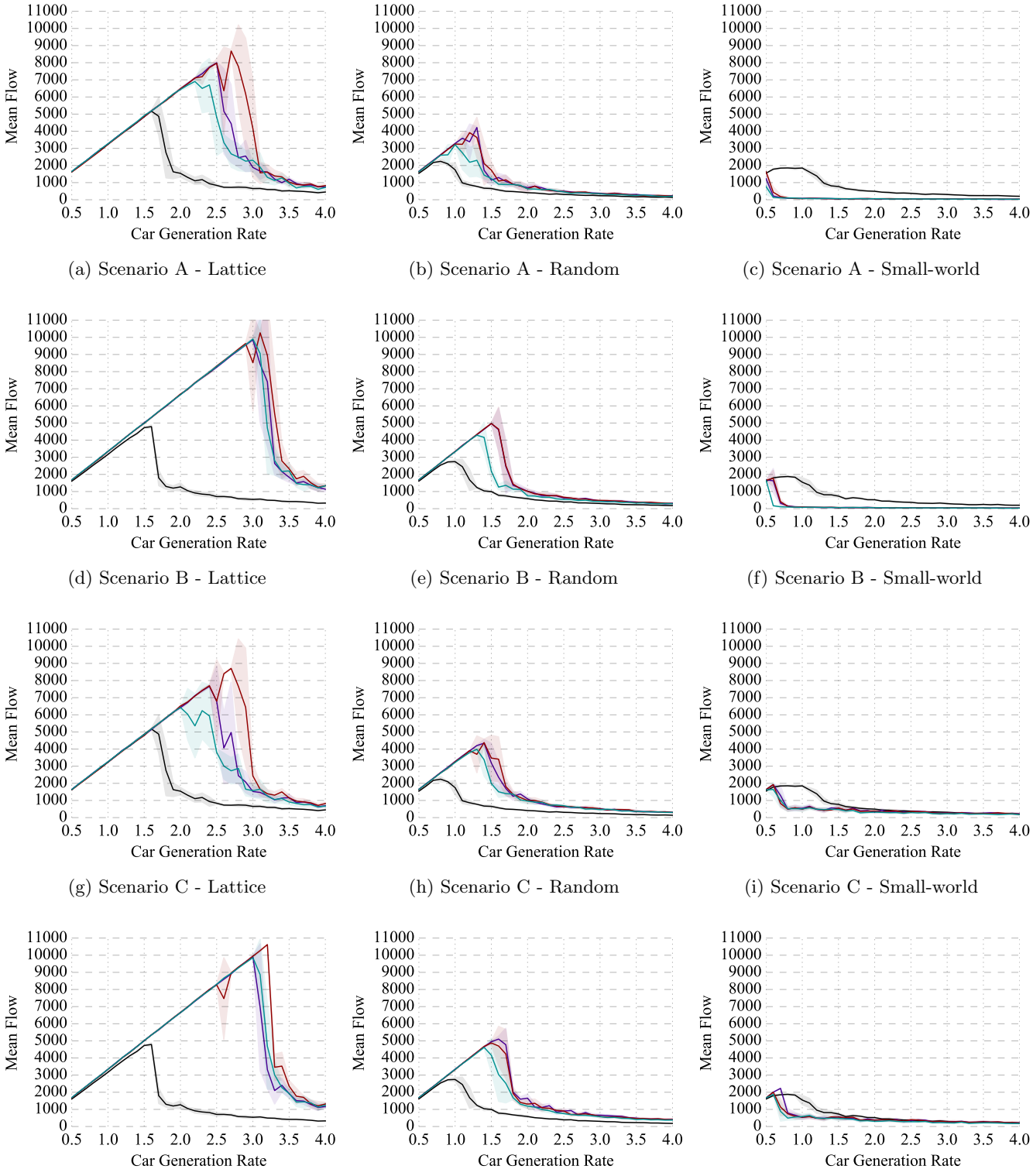
We find that our proposed intersection controller greatly improves the performance of the lattice and random networks, but that in the small-world network the fixed-interval traffic lights perform better (except for very low car generation rates). Specifically this relates to higher peak network flow, and shorter trip duration and waiting times.

6.2.1 Flow Through The Network

We find that whilst the network remains uncongested, the mean flow (in vehicles per hour) increases linearly with the car generation rate, up to some threshold value. Once the threshold value is reached the mean flow decreases as the car generation rate increases, indicating that the network has become congested and is unable to function properly (see Figure 6.3).

Lattice network

We find that in the fixed-interval case we reliably produce a peak flow through the network of around 5000 cars per hour when using 31 second intervals (Figures 6.3a and 6.3g), and



(a) Scenario A - Lattice (b) Scenario A - Random (c) Scenario A - Small-world

(d) Scenario B - Lattice (e) Scenario B - Random (f) Scenario B - Small-world

(g) Scenario C - Lattice (h) Scenario C - Random (i) Scenario C - Small-world

(j) Scenario D - Lattice (k) Scenario D - Random (l) Scenario D - Small-world

Figure 6.3: Car generation rate (vehicles per second) plotted against mean flow through the network (vehicles per hour). Results are shown for Scenarios A to D (by row) and three network topologies (by column). The *stage selection algorithm* and *stage duration* parameters are specified by the Scenario. The lines shown are for Tmin/Tmax (purple), Proportional (maroon) and Model Based (blue) stage duration algorithms. Fixed-cycle control is shown in black. Standard deviation between simulation runs is shown by the shaded region.

a slightly lower peak flow of around 4800 when using 15 second intervals (Figures 6.3d and 6.3j). Comparing the 31-second fixed-interval with the combination of capacity-aware stage selection and proportional stage duration in Scenario D, we see that the overall increase in peak flow is 110%.

Random network

We find that using a fixed-interval traffic light controller we reliably produce a peak flows through the network of around 2100 cars per hour when using 31 second intervals (Figures 6.3b and 6.3h), and a slightly higher peak flow of around 2700 when using 15 second intervals (Figures 6.3e and 6.3k). Comparing the 15-second fixed-interval with the combination of capacity-aware stage selection and Tmin/Tmax stage duration (Scenario D) demonstrates an 85% increase in maximum network flow during the simulation.

Small-world network

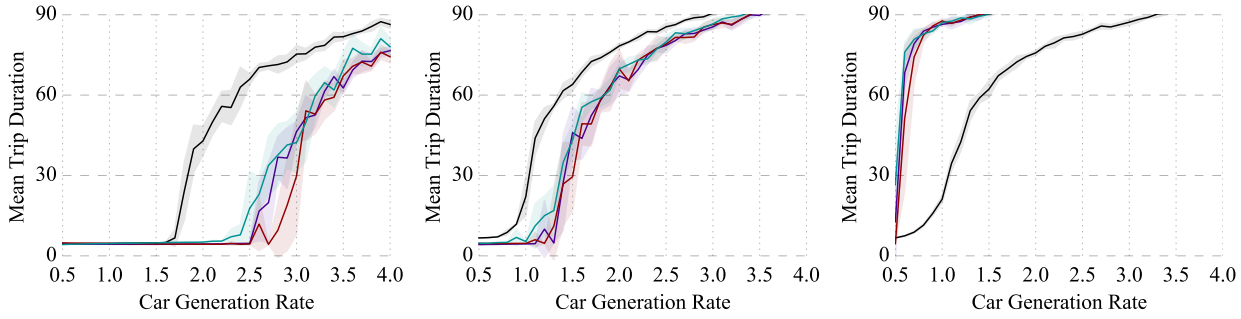
In the small-world network, we find that the maximum flows we can achieve are lower than in both the lattice and random networks. Specifically, the maximum network flow under a fixed-interval scheme is 1900 vehicles per hour, when using an interval of 31 seconds (Figures 6.4c and 6.3h) or 15 seconds (Figures 6.4f and 6.3l). Comparing either the 15 or 31-second fixed-interval schemes with the combination of capacity-aware stage selection and Tmin/Tmax stage duration (Scenario D) demonstrates a 16% increase in maximum network flow during the simulation. However, in these simulations, the mean flow dropped dramatically after the maximum flow was reached, and the fixed cycle controller demonstrated greater performance for all simulations above a car generation rate of 0.7.

6.2.2 Mean Trip Duration

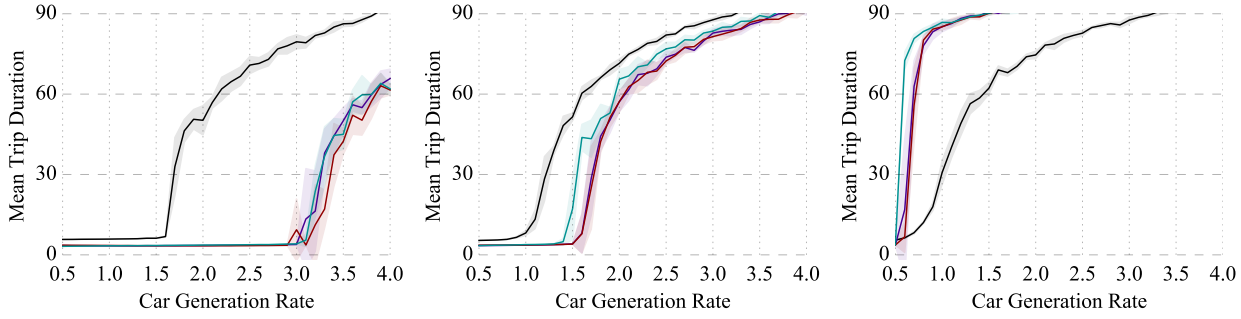
We find that the mean trip duration undergoes small variations as the car generation rate is varied, up to some critical car generation rate (see Figure 6.5). Past the critical car generation rate, we find large increases in the mean trip duration, indicative of congestion and gridlock in the network (see Figure 6.4). This corresponds to the reduction in flow seen in Figure 6.3. We find that the parameters of the stage duration algorithm impact on the mean trip duration, even at low car generation rates below the threshold value. Here we present the mean trip durations found prior to congestion, and the car generation rate at which we begin to observe very large trip durations.

Lattice network

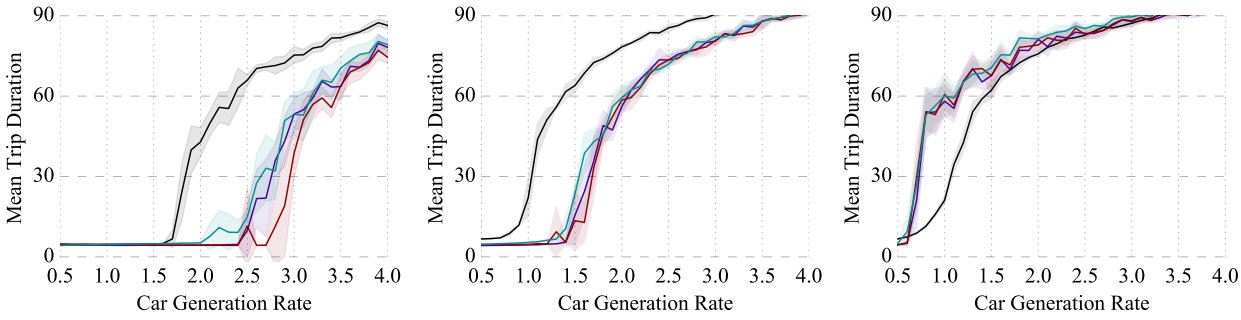
We find that for a fixed-interval scheme of 30 second intervals there is a mean trip duration of 4.5 to 5 minutes up to a car generation rate of 1.6 (Figures 6.4a and 6.5a). We observe that for a 15 second cycle there is a slightly higher mean trip duration of 5.75 to 6.75 minutes for car generation rates up to 1.6 (Figures 6.4d and 6.5d). Comparing the best



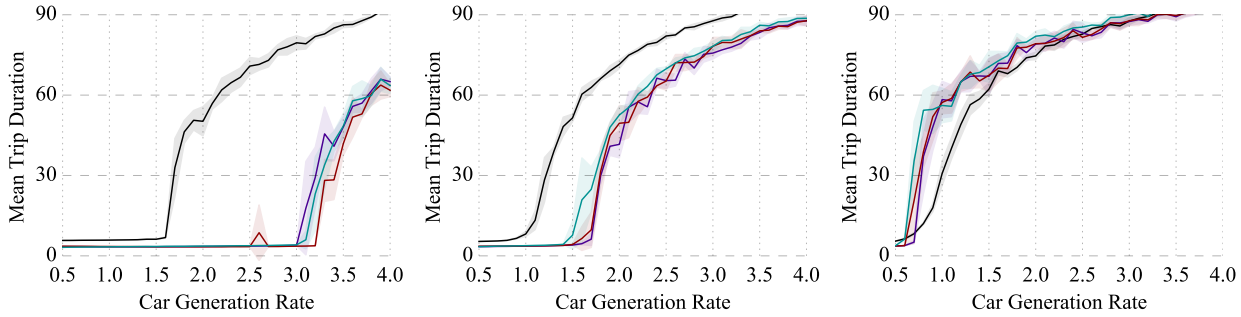
(a) Scenario A - Lattice (b) Scenario A - Random (c) Scenario A - Small-world



(d) Scenario B - Lattice (e) Scenario B - Random (f) Scenario B - Small-world

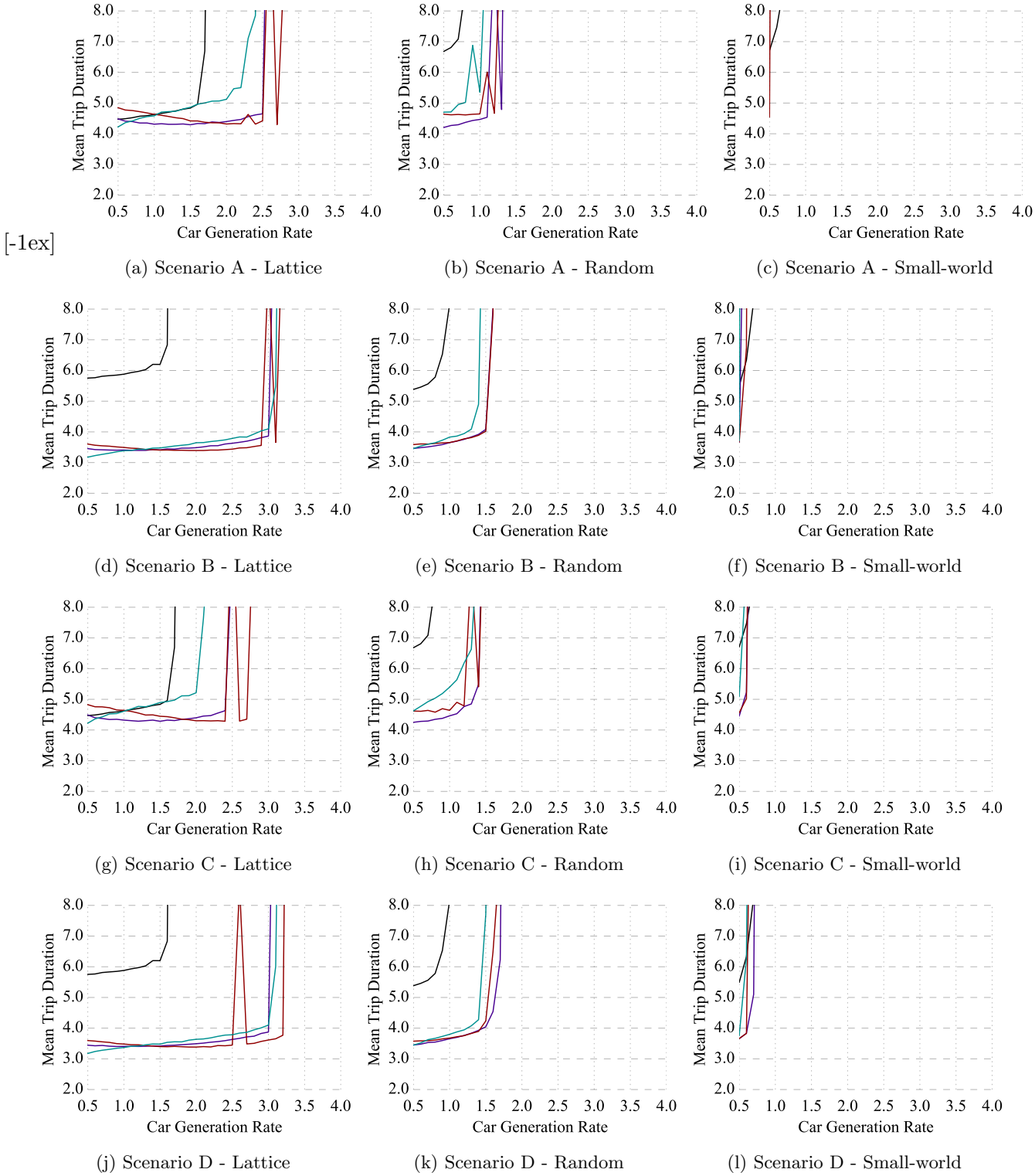


(g) Scenario C - Lattice (h) Scenario C - Random (i) Scenario C - Small-world



(j) Scenario D - Lattice (k) Scenario D - Random (l) Scenario D - Small-world

Figure 6.4: Car generation rate (vehicles per second) plotted against mean trip duration (minutes). Results are shown for Scenarios A to D (by row) and three network topologies (by column). The *stage selection algorithm* and *stage duration* parameters are specified by the Scenario. The lines shown are for Tmin/Tmax (purple), Proportional (maroon) and Model Based (blue) stage duration algorithms. Fixed-cycle control is shown in black. Standard deviation between simulation runs is shown by the shaded region.



[-1ex]

(a) Scenario A - Lattice (b) Scenario A - Random (c) Scenario A - Small-world

(d) Scenario B - Lattice (e) Scenario B - Random (f) Scenario B - Small-world

(g) Scenario C - Lattice (h) Scenario C - Random (i) Scenario C - Small-world

(j) Scenario D - Lattice (k) Scenario D - Random (l) Scenario D - Small-world

Figure 6.5: Close-Up of Figure 6.4 showing performance differences at car generation rates below the critical threshold. Car generation rate (vehicles per second) plotted against mean trip duration (minutes). Results are shown for Scenarios A to D (by row) and three network topologies (by column). The *stage selection algorithm* and *stage duration* parameters are specified by the Scenario. The lines shown are for Tmin/Tmax (purple), Proportional (maroon) and Model Based (blue) stage duration algorithms. Fixed-cycle control is shown in black. Standard deviation between simulation runs is shown by the shaded region.

performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions we reduced tip duration by 22-25 %.

Random network

We find that for a fixed-interval scheme of 30 seconds there is a mean trip duration of 6.5 to 7 minutes up to a car generation rate of 0.7 (Figures 6.4b and 6.5b). We observe that for a 15 second cycle there is a slightly lower mean trip duration of 5.5 to 6.5 minutes for car generation rates up to 0.9 (Figures 6.4e and 6.5e). Comparing the best performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions we reduced trip duration by 36-38%.

Small-world network

We find that for a fixed-interval scheme of 30 seconds there is a mean trip duration of 6.75 to 7.5 minutes up to a car generation rate of 0.6 (Figures 6.4c and 6.5c). We observe that for a 15 second cycle there is a slightly lower mean trip duration of 5.5 to 6.25 minutes for car generation rates up to 0.6 (Figures 6.4f and 6.5f). Comparing the best performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions below a car generation rate of 0.7 we reduced tip duration by 32-36%. However, past this point, the benefit is lost, and the network becomes congested creating high trip durations than for a fixed-interval scheme.

Summary of mean trip duration results

In all simulations, there were only minor differences observed due to the Stage Selection Algorithm. In contrast, the different Stage Selection Algorithms exhibited individual behaviours at car generation rates below the critical rate. In the lattice network, we observe that the model based controller usually exhibits the lowest trip duration for the smallest car generation rates and that the trip duration increases with the car generation rate. The proportional algorithm behaves quite differently, giving higher trip durations for lower car generation rates, which decrease and then increase again as the critical car generation rate is approached (see Figure 6.5j). The Tmin/Tmax algorithm provides trip durations somewhere in between the two.

This behaviour may be explained due to the responsiveness of the algorithms. The model based algorithm predicts the stage duration before it begins, whereas the other algorithms are adaptive based on historical performance. If the initial stage duration for the Tmin/Tmax and proportional algorithms is too low or high for the car generation rate, it will take some time to adapt. It will also perform best when it starts the simulation with a stage duration which is approximately optimum for the level of traffic in the simulation (probably some car generation rate between 0 and the critical rate).

6.2.3 Mean Waiting Time

Figures 6.6 and 6.7 show the mean waiting time for vehicles in the network. The waiting time corresponds to the mean trip time results in each network, and increases in the waiting time are most likely the biggest contributor to increases in trip time.

Lattice network

We find that for a fixed-interval scheme of 30 seconds there is a mean wait time of 2.25 to 2.75 minutes up to a car generation rate of 1.6 (Figures 6.6a and 6.7a). We observe that for a 15 second cycle there is a slightly higher mean wait time of 3.25 to 4.25 minutes for car generation rates up to 1.6 (Figures 6.6d and 6.7d). Comparing the best performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions we reduced wait time by 56-54%.

Random network

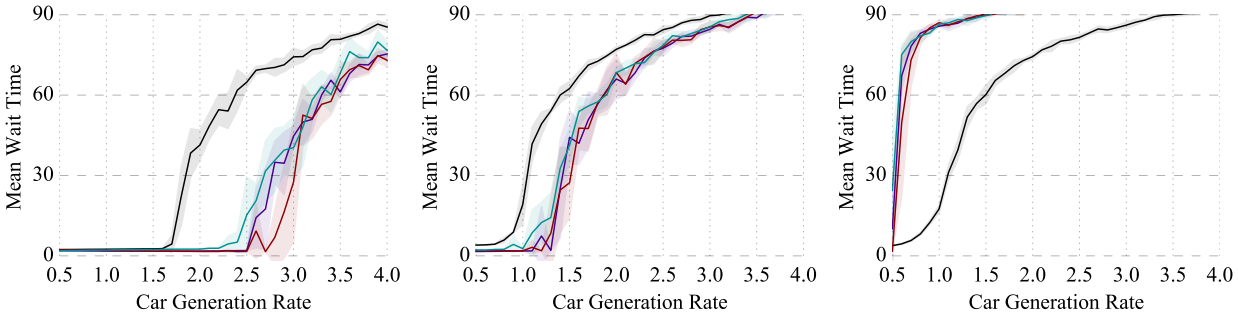
We find that for a fixed-interval scheme of 30 seconds there is a mean wait time of 4.0 to 4.25 minutes up to a car generation rate of 0.7 (Figures 6.6b and 6.7b). We observe that for a 15 second interval there is a slightly lower mean wait time of 2.75 to 3.75 minutes for car generation rates up to 0.9 (Figures 6.6e and 6.7e). Comparing the best performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions we reduced wait time by 64-67%.

Small-world network

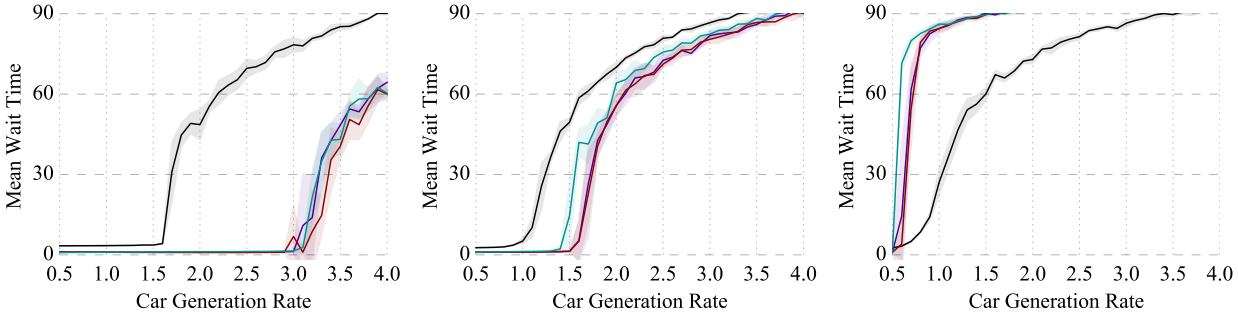
We find that for a fixed cycle scheme of 30 seconds there is a mean wait time of 3.75 to 4.5 minutes up to a car generation rate of 0.6 (Figures 6.6c and 6.7c). We observe that for a 15-second cycle there is a slightly lower mean wait time of 2.75 to 3.25 minutes for car generation rates up to 0.6 (Figures 6.6f and 6.7f), but this also increases sharply past this point. Comparing the best performing fixed-interval scheme with our best performing intersection controller, we find that in uncongested conditions we reduced wait time by 64-69%. However, this reduction is mitigated when we pass the critical car generation rate for our intersection controller.

Summary of mean waiting time results

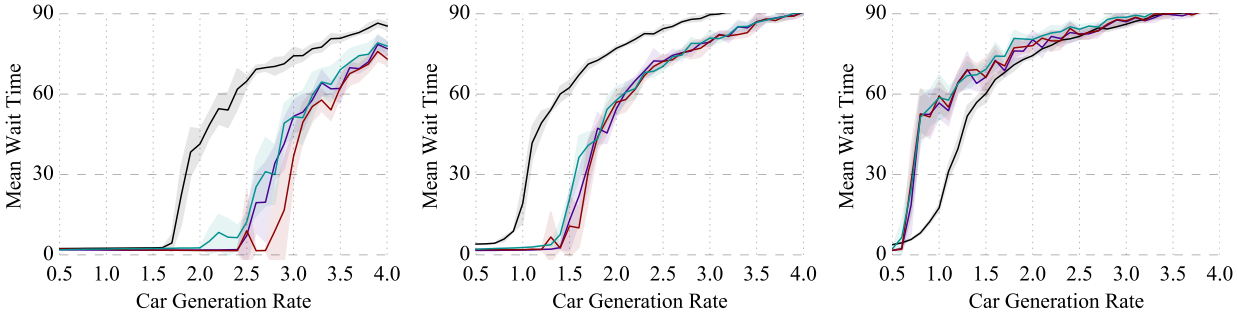
Qualitatively the mean wait time for fixed-interval and our various intersection controllers matched the mean trip duration. This result is unsurprising, as reduced stopping should result in reduced travel time.



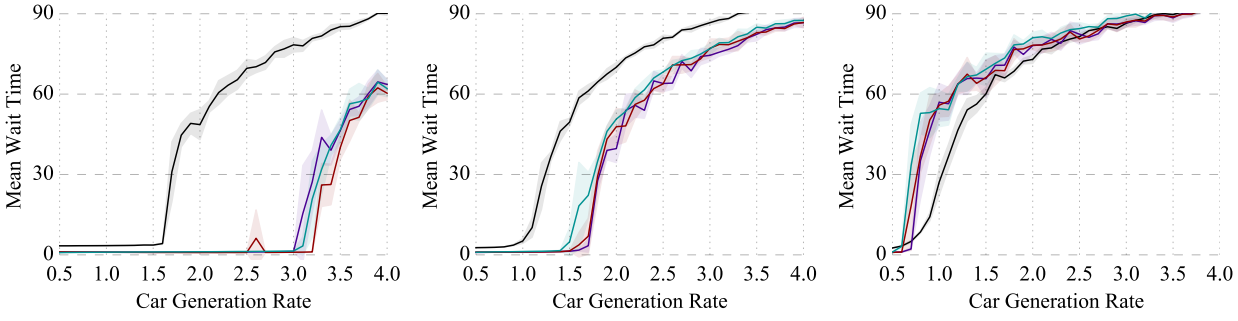
(a) Scenario A - Lattice (b) Scenario A - Random (c) Scenario A - Small-world



(d) Scenario B - Lattice (e) Scenario B - Random (f) Scenario B - Small-world

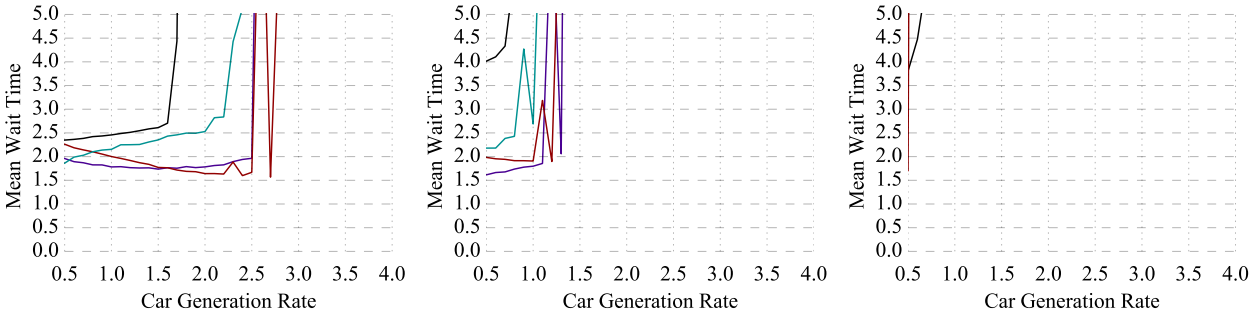


(g) Scenario C - Lattice (h) Scenario C - Random (i) Scenario C - Small-world

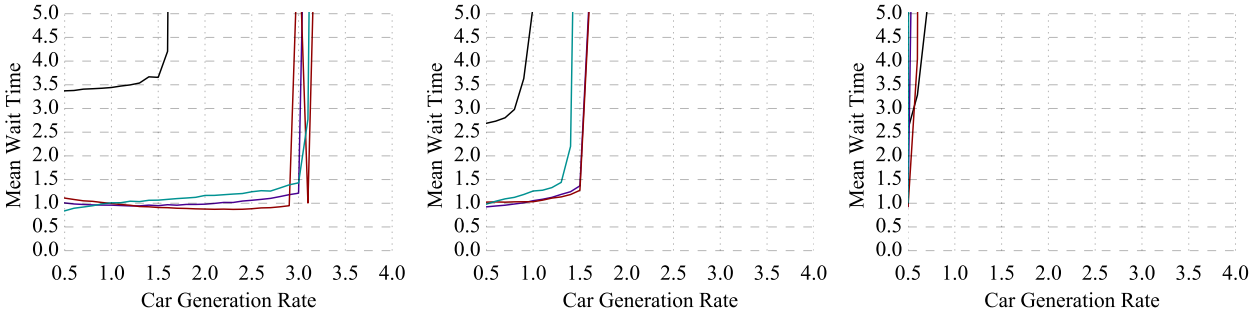


(j) Scenario D - Lattice (k) Scenario D - Random (l) Scenario D - Small-world

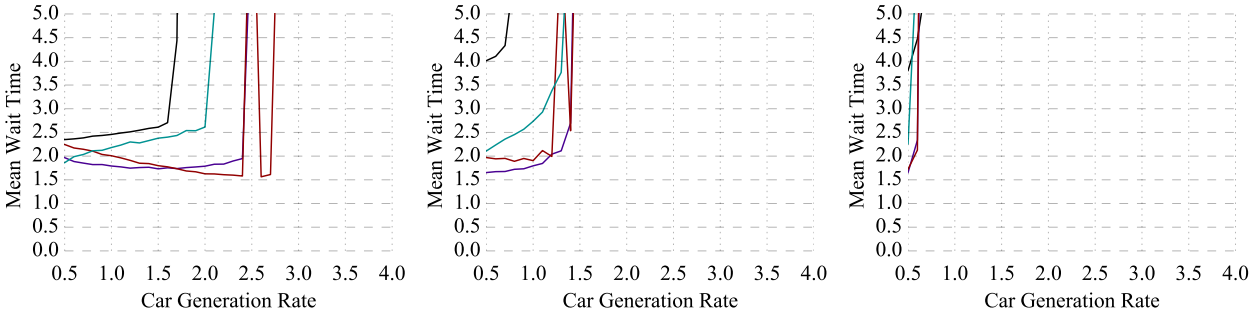
Figure 6.6: Car generation rate (vehicles per second) plotted against mean wait time (minutes). Results are shown for Scenarios A to D (by row) and three network topologies (by column). The *stage selection algorithm* and *stage duration* parameters are specified by the Scenario. The lines shown are for Tmin/Tmax (purple), Proportional (maroon) and Model Based (blue) stage duration algorithms. Fixed-cycle control is shown in black. Standard deviation between simulation runs is shown by the shaded region.



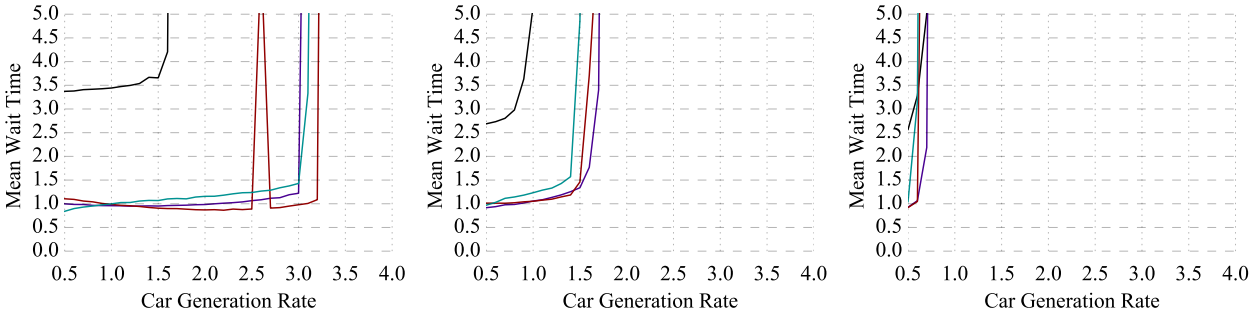
(a) Scenario A - Lattice (b) Scenario A - Random (c) Scenario A - Small-world



(d) Scenario B - Lattice (e) Scenario B - Random (f) Scenario B - Small-world



(g) Scenario C - Lattice (h) Scenario C - Random (i) Scenario C - Small-world



(j) Scenario D - Lattice (k) Scenario D - Random (l) Scenario D - Small-world

Figure 6.7: Close-up images of Figure 6.6 showing in more detail the performance of the algorithms when the mean wait time is less than 5 minutes (which we consider to mean the network is not congested).

Table 6.3: Summary of controller performance in the synthetic scenarios. The mean trip duration and mean wait time are taken for results below the critical car generation rate when the network remains uncongested. The best results in the lattice network were gained when using the capacity-aware stage selection algorithm and the proportional stage duration algorithm in Scenario D. The best results in the random network were gained from the congestion-aware stage selection algorithm and the Tmin/Tmax stage duration algorithm in Scenario B.

	Fixed-Interval		**Work-Conserving**											
	Interval		**Tmin/Tmax**				**Proportional**				**Model Based**			
Metric	15	31	A	B	C	D	A	B	C	D	A	B	C	D
Lattice														
Peak Flow	4800	5000	8000	9800	8000	10000	8700	10250	8700	10500	7000	9800	7000	10000
Mean Trip Duration	5.75-6.75	4.5-5.0	4.25-4.5	3.5-3.75	4.25-4.5	3.5-4.0	4.25-4.75	3.5	4.25-4.75	3.5-3.75	4.25-5.5	3.25-4.0	4.25-5.25	3.25-4.0
Mean Wait Time	3.25-4.25	2.25-2.75	1.75-2.0	1.0-1.25	1.75-2.0	1.0-1.25	1.75-2.25	1.0	1.5-2.25	1.0	2.0-3.0	1.0-1.5	2.0-2.5	1.0-1.5
Crit. Rate	1.6	1.6	2.5	3.0	2.4	3.0	2.7	3.1	2.7	3.2	2.2	3.0	2.0	3.0
Random														
Peak Flow	2700	2100	4100	5000	4200	4900	4000	5000	4200	5000	3000	4200	4000	4600
Mean Trip Duration	5.5-6.5	6.5-7.0	4.25-4.75	3.5-4.0	4.25-4.75	3.5-4.0	4.5-4.75	3.5-4.0	4.5-4.75	3.5-4.0	4.5-5.5	3.5-4.0	4.5-5.5	3.5-4.25
Mean Wait Time	2.75-3.75	4.0-4.25	1.5-2.0	1.0-1.25	1.5-2.0	1.0-1.25	2.0	1.0-1.25	2.0	1.0-1.25	2.0-2.75	1.0-1.5	2.0-3.0	1.0-1.5
Crit. Rate	0.9	0.7	1.3	1.5	1.3	1.5	1.2	1.5	1.2	1.4	1.0	1.3	1.1	1.4
Small-world														
Peak Flow	1900	1900	1100	1000	1900	2200	1600	1000	2000	1900	900	1000	1600	1900
Mean Trip Duration	5.5-6.25	6.75-7.5	12.75	3.75	4.5-5.0	3.75-4.0	4.5	3.75	4.5-5.0	3.75-4.0	30	3.75	5.0	3.75
Mean Wait Time	2.75-3.25	3.75-4.5	10.25	1.0	1.75-2.25	1.0	1.75	1.0	1.75-2.25	1.0	25	1.0	2.25	1.0
Crit. Rate	0.6	0.6	0.5	0.5	0.6	0.6	0.5	0.5	0.6	0.6	0.5	0.5	0.5	0.5

Table 6.4: Flow comparison between best fixed cycle and best controlled simulation in synthetic networks.

Network	Fixed-Interval (Interval)	Work-Conserving (Stage Duration - Scenario)	% Change
Lattice	5000 (31)	10500 (Proportional - D)	+110%
Random	2700 (15)	5000 (Tmin/Tmax or Proportional - B)	+85%
Small-world	1900 (15)	2200 (Tmin/Tmax - D)	+16%

Table 6.5: Mean trip duration comparison between best fixed cycle and best controlled simulation in synthetic networks.

Network	Fixed-Interval (Interval)	Work-Conserving (Stage Duration - Scenario)	% Change
Lattice	4.5-5 (31)	3.5-3.75 (Proportional - D)	-22% to -25%
Random	5.5-6.5 (15)	3.5-4.0 (Tmin/Tmax or Proportional - B)	-36% to -38%
Small-world	5.5-6.25 (15)	3.75-4.0 (Tmin/Tmax - D)	-32% to -36%

6.2.4 Discussion of Congestion-Aware and Capacity-Aware Algorithm Performance

The decentralised intersection controller shows clear improvement over the fixed interval scheme in the scenarios tested, see Tables 6.4 to 6.6 for a summary of the improvements. In the lattice and random networks, we find that a fixed cycle is outperformed by our intersection controller both in terms of the maximum flow in the network and in reducing the mean trip duration. In the small-world network, we find that an improvement is only observed for very low car generation rates below 0.6 vehicles per second. At higher car generation rates all controlled and uncontrolled schemes deteriorate in performance, however, the controlled schemes do this at a faster rate.

We observe that in all the synthetic networks, the best results were achieved with parameters that encouraged shorter stage durations (scenarios B and D - see Table 6.3). In particular, the flow in the lattice network is maximised when combining the proportional stage duration with capacity-aware stage selection algorithm, and parameters from scenario D. In the random network the best performance is found when combining the proportional or Tmin/Tmax stage duration algorithm with the congestion-aware stage selection algorithm, and parameters from scenario B.

Shorter stage durations offer the advantage of allowing the intersection to more fre-

Table 6.6: Mean wait time comparison between best fixed cycle and best controlled simulation in synthetic networks.

Network	Fixed-Interval (Interval)	Work-Conserving (Stage Duration - Scenario)	% Change
Lattice	2.25-2.75 (31)	1.0 (Proportional - D)	-56% to -64%
Random	2.75-3.75 (15)	1.0-1.25 (Tmin/Tmax or Proportional - B)	-64% to -67%
Small-world	2.75-3.25 (15)	1.0 (Tmin/Tmax - D)	-64% to -69%

Table 6.7: Critical car generation rate comparison between best fixed cycle and best controlled simulation in synthetic networks.

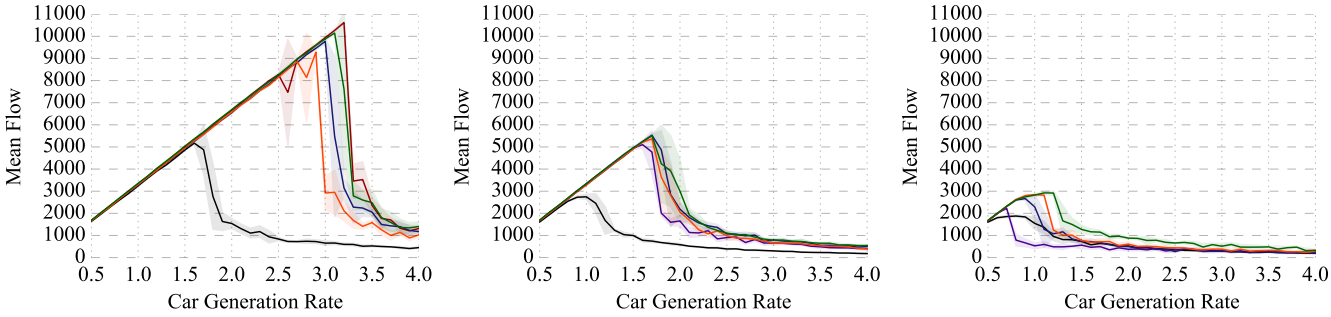
Network	**Fixed-Interval (Interval)**	**Work-Conserving (Stage Duration - Scenario)**	**% Change**
Lattice	1.6 (31)	3.2 (Proportional - D)	+100%
Random	0.9 (15)	1.5 (Tmin/Tmax - D)	+67%
Small-world	0.6 (15)	0.6 (Tmin/Tmax - D)	0%

quently select the next stage. Once a queue has been emptied, the next stage should be selected as quickly as possible in order to maintain work-conservation at the intersection. If a full queue can be emptied somewhere between 5 and 25 seconds, then in Scenarios B and D this value will be settled on faster.

Also, we find that the network topology has a clear impact on the performance of the network under all controlled and uncontrolled schemes. In the lattice network, we observe a much higher network flow than in the random network. This result is not surprising when we compare this with the betweenness centrality of the networks (Figure 6.2). We observe that the lattice network has many nodes with a relatively even betweenness centrality, but in the random network, the shortest paths are concentrated through only a few nodes. The flow in the network is clearly related to the betweenness centrality, as networks with many possible shortest paths will be less prone to congestion.

The mean trip duration clearly correlates with the network flow. When the network becomes congested and the flow stops increasing in proportion to the increasing demand, we see simultaneous drops in network flow and increases in the mean trip time. Similarly, we find that the mean waiting time directly correlates with the mean trip time, showing that reducing waiting time at traffic lights will have a direct impact on travel times. We term the car generation rate at which the mean trip duration increases non-linearly in relation to the car generation rate a 'critical' car generation rate (see Table 6.7).

In many of the Figures, we observe some oscillations in the results. Using Figure 6.3a as an example, we see the proportional stage duration scheme sees a dip in the mean flow between car generation rates of 2.6 and 2.8, and that past 2.8 the mean flow drops significantly. We also see that there is a corresponding increase in the standard deviation between simulations runs at this point. This indicates that at values near to the critical car generation rate there is some instability between simulation runs and that different combinations of routes may induce or avoid congestion in the network (which is responsible for the reduction in mean flow and increase in trip duration). As further evidence of this, we see that as the car generation rates increase away from the threshold value, the standard deviation between runs decreases.



(a) Lattice (b) Random (c) Smallworld

Figure 6.8: Car generation rate (vehicles per second) plotted against mean flow through the network (vehicles per hour) in the 3 network topologies. Intersection controllers shown are OBB (dark blue), Gregoire et. al. (orange), and PPC (dark green). For comparison we include the 15 second fixed-cycle controller, and the best performing controller from our results in section 6.2 (this is either the proportional (maroon) or Tmin/Tmax (purple) stage duration algorithm from Scenario D depending on the network topology).

6.3 Numerical Analysis of the Pressure Propagating Controller

The PPC was tested alongside two decentralised backpressure algorithms based on the current literature. A backpressure controller uses the difference in length between the queue on the input link and the queue on the output link to calculate the "pressure" of any particular phase. Stages are selected by minimising pressure at the intersection.

The backpressure algorithms selected were:

Occupancy Based Backpressure (OBB) , which is a backpressure controller where queue length on a link is measured as the occupancy of the road.

Gregoire et. al. [46] which is congestion-aware backpressure controller. The Gregoire et. al. algorithm is similar to the OBB algorithm, however, it has modifications which make it less susceptible to deadlocks.

The PPC requires all intersections to be synchronised for the propagation stage, and so a fixed stage duration was used for analysis. The same stage duration was used for all controllers tested here.

We chose the Gregoire et. al. algorithm because it is the first back-pressure algorithm in the literature (to our knowledge) that takes into account the effect of downstream congestion and uses a normalised back pressure to avoid gridlock situations that are possible with some algorithms.

Results for network flow, trip duration, and wait time are found in Figures 6.8, 6.9 and 6.10.

6.3.1 Flow Through The Network

In Figure 6.8 we observe the mean flow in the network for these simulations.

Lattice network

In the lattice network (Figure 6.8a) the PPC achieves a peak mean flow of around 10150 vehicles per hour, at a car generation rate of 3.1. This outperforms the OBB and Gregoire algorithms which achieve peaks of 9750 and 9300 respectively. This is a slightly lower performance than the best controller tested in the previous section. In those simulations, the capacity-aware stage selection combined with proportional stage duration achieved a peak flow of 10600 vehicles per hour at a car generation rate of 3.2. All of the algorithms outperform a fixed-cycle controller by a minimum of 81%.

Random network

In the random network (Figure 6.8b) we find that the performance of the PPC, OBB and Gregoire algorithms are similar, although the best performance is given by OBB which achieves a peak flow through the network of 5500 vehicles per hour. The 3 algorithms outperform not only the fixed-cycle controller but also the best capacity-aware controller we tested in the random network. In this case the OBB increases flow 9% over our capacity-aware controller but is only giving an improvement of 1% over the PPC.

Small-world network

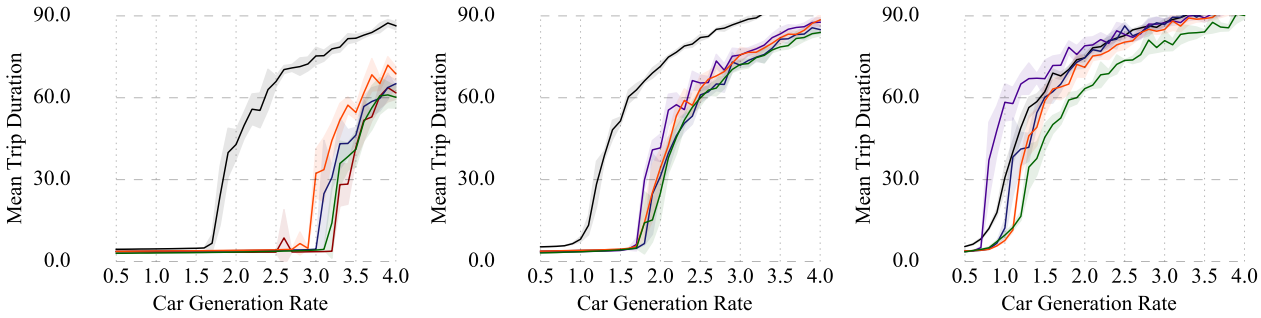
In the small-world network (Figure 6.8c the PPC shows the best performance of all of the algorithms tested. The PPC achieves a peak flow of 2900 vehicles per hour. The Gregoire algorithm performs comparably well but falls short at 2850 vehicles per hour. The maximum flow achieved by the PPC represents a 29% increase in peak flow over that achieved using our best capacity-aware algorithm tested previously and 53% over the fixed-cycle controller.

Summary of flow results

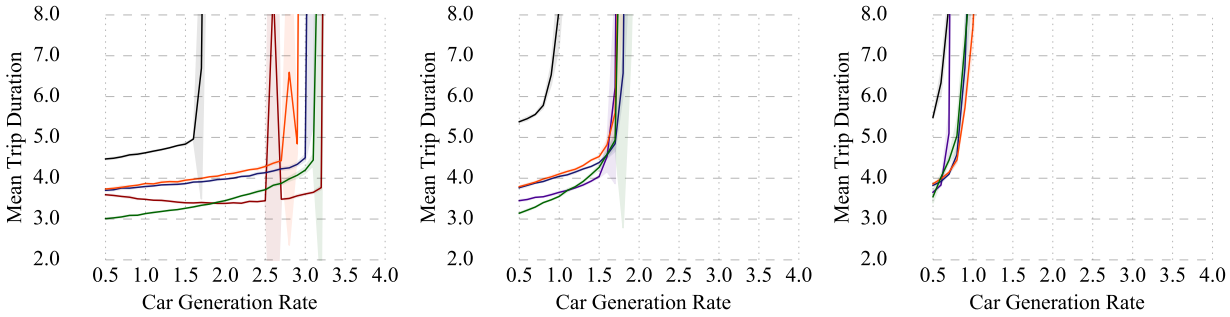
The backpressure algorithms show increased flow in the random and small-world networks over the algorithms tested previously, but our previous algorithms perform better in the lattice network. Unlike the lattice network, the random and small-world networks consist of a mix of road lengths. The Gregoire algorithm is designed to perform better in networks with a mix of road lengths, and the algorithm clearly works. The design of our PPC algorithm seems to further increase traffic flow in these networks.

6.3.2 Mean Trip Duration

In Figure 6.9 we see mean trip duration for the simulations. As in our previous analysis, we find that an increase in trip duration corresponds to a sudden decrease in flow through the network. In contrast to the previous intersection controllers, we observe a strictly monotonically increasing trip duration in relation to increasing car generation rate. We have used the decrease in network flow as an indicator of congestion in the network, and we comment on the car generation rate at which this occurs, as well as commenting on trip duration.



(a) Lattice (b) Random (c) Smallworld



(d) Lattice (e) Random (f) Small-world

Figure 6.9: Car generation rate (vehicles per second) plotted against mean trip duration (minutes). Intersection controllers shown are OBB (dark blue), Gregoire et. al. (orange), and PPC (dark green). For comparison we include the best performing fixed-cycle controller, and the best performing controller from our results in section 6.2 (this is either the proportional (maroon) or Tmin/Tmax (purple) stage duration algorithm from Scenario D depending on the network topology).

Lattice network

In the lattice network (Figures 6.9a and 6.9d) we find that the PPC achieves a mean trip duration of 3 minutes at the lowest car generation rate. This is the lowest mean trip duration achieved by any algorithm tested here. Comparing the mean trip duration for the PPC to that of our capacity-aware controller for car generation rates below 3.2 (i.e. prior to the onset of congestion), we find that the PPC gives the lowest mean trip duration up to a car generation rate of approximately 1.8. Past this value, the capacity-aware algorithm provides a lower mean trip duration. The OBB and Gregoire algorithms still outperform the fixed-cycle controller, reducing the mean trip duration by up to 1 minute, however, they are outperformed by our own algorithms in this network.

Random network

In the random network (Figures 6.9b and 6.9e) we again observe that the PPC provides the lowest mean trip duration for any of the algorithms for the lowest car generation rates. The PPC gives a mean trip duration of 3.25 minutes, which is 0.25 minutes less than that for the next best performing algorithm, OBB. All of our algorithms again outperform fixed-cycle control, and aside from small variations in trip duration during uncongested conditions the performance of our algorithms is comparable.

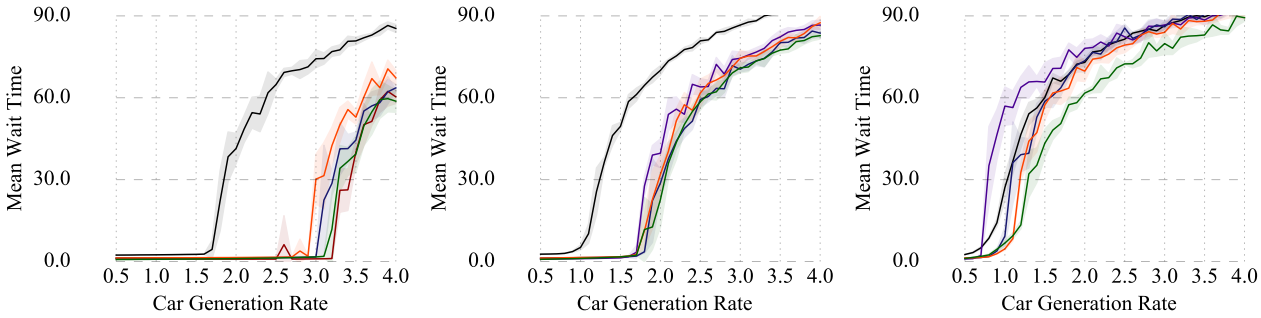
Small-world network

In the small-world network (Figures 6.9c and 6.9f) we find that the trip duration increases quickly with car generation rate for all algorithms. At the lowest car generation rate tested the OBB and PPC provide the lowest mean trip duration around 3.5 minutes, however, similar results are found for the other algorithms. In Figure 6.9c we observe that our capacity-aware algorithm from the previous section is easily outperformed by the PPC, OBB, and Gregoire algorithms. Specifically, the PPC outperforms the fixed-cycle controller in terms of a lower mean trip duration for all car generation rates.

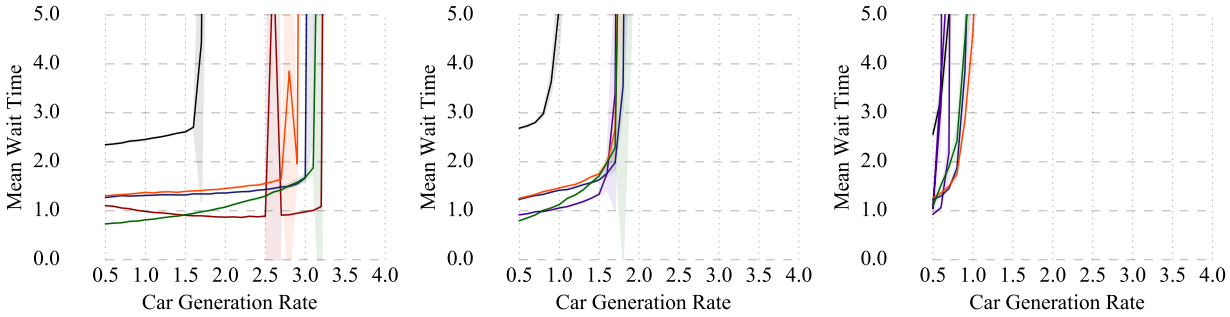
Summary of mean trip duration results

We observe that when compared with algorithms in the previous section, only the PPC provides any performance benefits in the lattice network. Specifically, PPC provides the lowest mean trip duration for car generation rates below 1.8. In the random network, we observe an even stronger improvement of both the PPC and OBB algorithms over our previous capacity-aware controller in terms of mean trip duration.

Finally, we find that in the small-world network all three algorithms are able to outperform our capacity-aware algorithm. We find that at some car generation rates below a value of 1 the PPC appears to be outperformed by the Gregoire algorithm, but that at others it outperforms them. In Figure 6.9c we see that in the long term PPC has the lowest mean trip duration.



(a) Lattice (b) Random (c) Small-world



(d) Lattice (e) Random (f) Small-world

Figure 6.10: Car generation rate (vehicles per second) plotted against mean wait time (minutes). Intersection controllers shown are OBB (dark blue), Gregoire et. al. (orange), and PPC (dark green). For comparison we include the best performing fixed-cycle controller, and the best performing controller from our results in section 6.2 (this is either the proportional (maroon) or Tmin/Tmax (purple) stage duration algorithm from Scenario D depending on the network topology).

6.3.3 Mean Waiting Time

In Figure 6.10 we can see the mean wait time for the simulations. As was observed in the simulations in the previous section, we find that wait time correlates almost exactly with trip duration.

Lattice network

In the lattice network (Figures 6.10a and 6.10d) we find that the mean wait time correlates strongly with the mean trip duration. As with mean trip duration, we find that the best performance is given by the PPC at the lowest car generation rate, where the mean wait time is around 0.75 minutes. This reflects a reduction in wait time to around 33% of that obtained using the fixed-cycle controller. Between a car generation rate of 1.5 and 3.2 our previous capacity-aware algorithm gives the lowest mean wait time in the network. The PPC and our capacity-aware algorithm beat both the OBB and Gregoire algorithms.

Random network

In the random network (Figures 6.10b and 6.10e) we observe that the PPC gives the lowest mean wait time at the lowest car generation rate, however the OBB gives the lowest mean wait time for car generation rates between 0.8 and 1.6. All of our algorithms outperform fixed-cycle control.

Small-world network

In the small-world network (Figures 6.10c and 6.10f) we find that the lowest mean wait times are achieved by the PPC and the Gregoire algorithms. In comparison to the fixed-cycle scheme we find that all algorithms reduce wait time by at least 50% at the lowest car generation rate, when compared to the fixed-cycle scheme, however only the new algorithms presented here (OBB, Gregoire and PPC) are able to consistently beat the fixed-cycle at all car generation rates.

Summary of mean waiting time results

In general, we find that that mean trip duration correlates with trip duration for all of the algorithms. An interesting property of the PPC can be observed by studying Figures 6.9d and 6.10d. We find that the wait time forms a larger part of trip duration in the PPC than our capacity-aware controller (this can easily be observed by noting that the green and maroon lines cross at different car generation rates in the two figures). This indicates that the average road speed is higher using the PPC, and the wait times for some vehicles must be pushed higher due to this.

6.4 Discussion

In Tables 6.8, 6.9, 6.10 and 6.11 we compare the performance metrics for all of the algorithms presented in this chapter. We use the best case of fixed-interval control and

capacity-aware control for comparison.

The results indicate that the peak capacity of a network can be increased and the mean trip time reduced using decentralised traffic-responsive intersection control algorithms, rather than fixed-cycle schemes.

The results also show that different algorithms may be appropriate for use in different network topologies and with different aims for the network. For example, in the small-world network, the Gregoire and PPC control outperformed all other algorithms in terms of maximum flow through the network, mean trip duration, and mean wait time. In the lattice network, we observed the highest flow in the network using a capacity-aware controller, but we could achieve shortest trip durations for some car generation rates using the PPC.

These results indicate that an optimal control strategy might include a combination of control algorithms, which are switched according to the network topology and/or traffic conditions.

In the next chapter, we present a preliminary investigation into the performance of these controllers when simulated in a real-world scenario of the city of Luxembourg.

Table 6.8: Flow comparison between OBB, Gregoire, PPC, best fixed-interval and best Capacity-Aware intersection control

Network	**OBB**	**Gregoire**	**PPC**	**Fixed-Interval (best)**	**Capacity-Aware (best)**
Lattice	9770	9282	10147	5172	10618
Random	5529	5364	5513	2747	5104
Small-world	2665	2845	2923	1878	2220

Table 6.9: Trip duration comparison between OBB, Gregoire, PPC, best fixed-interval and best Capacity-Aware intersection control

Network	**OBB**	**Gregoire**	**PPC**	**Fixed-Interval (best)**	**Capacity-Aware (best)**
Lattice	3.75-4.5	3.75-4.5	3-4.5	4.5-5	3.5-3.75
Random	3.75-4.5	3.75-4.5	3-4.5	5.5-6.5	3.5-4
Small-world	3.9-12	3.5-12	3.9-14	5.5-6.25	3.75-4

Table 6.10: Mean wait time comparison between OBB, Gregoire, PPC, best fixed-interval and best Capacity-Aware intersection control

Network	**OBB**	**Gregoire**	**PPC**	**Fixed-Interval (best)**	**Capacity-Aware (best)**
Lattice	1.25-1.5	1.25-2	0.75-1.75	2.25-2.75	1
Random	1.25-2	1.25-2.5	0.75-2.25	2.75-3.75	1-1.25
Small-world	1.25-9.25	1.25-8.5	1-10	2.75-3.25	1

Table 6.11: Critical car generation rate comparison between OBB, Gregoire, PPC, best fixed-interval and best Capacity-Aware intersection control

Network	**OBB**	**Gregoire**	**PPC**	**Fixed-Interval (best)**	**Capacity-Aware (best)**
Lattice	2.9	3	3.1	1.6	3.2
Random	1.7	1.7	1.7	0.9	1.5
Small-world	0.8	1.1	1.1	0.6	0.6

Chapter 7

Preliminary Numerical Validation of Intersection Control Algorithms in a Real-World Scenario

The results from validation of the intersection control strategies across differing synthetic network topologies in Chapter 6 showed that network topology and loading had a significant impact on the performance of the algorithms. Here we look at a real-world scenario, with time-varying traffic loads a topology based on the road layout of the city of Luxembourg.

We perform a similar analysis to that found in the previous chapter, firstly testing a combination congestion-aware and capacity-aware stage selection algorithms with various stage duration algorithms and various parameter combinations. We compare these results with the Occupancy Based Backpressure (OBB), Gregoire, and Pressure Propagating Controller (PPC) algorithms. The results presented are to be taken as a preliminary investigation of the controller's performance on a more realistic network and under more realistic loads, and are therefore subject to far wider deviations than in the synthetic networks.

7.1 Luxembourg Scenario

The Luxembourg SUMO Traffic (LuST) Scenario is a scenario based on the city of Luxembourg. It was created by Codeca et al [24] (`https://github.com/lcodeca/LuSTScenario`). The scenario depicts 24 hours of transport in the city of Luxembourg. The network includes traffic light locations, roundabouts, a bypass around the city, as well as residential streets. Further details of the topology are given in Table 7.1. The scenario includes 24 hours of simulated traffic demands, which include through traffic (i.e. highway only traffic), internal traffic, and traffic travelling between internal and external locations (entering or exiting at the edges of the network). The traffic demand was generated using the ACTIVITYGEN program which is included as part of the traffic simulator Sumo. ACTIVITYGEN takes information about the network, such as work and residential zones, and

generates traffic demands using an activity-based traffic generation model. The scenario also includes bus routes and a mix of vehicles types. The demand has observable peaks during rush hours, consistent with expected traffic demands.

We use an 11-hour subset of the full demand for the purposes of our simulations (see Figure 7.3). Unlike in our synthetic networks, where the car generation rate was constant, in these simulations the car generation rate varies. The mean car generation rate throughout the entire simulation is less than 2.5 vehicles per time step, but when taking into account shorter time periods (such as between 8-9am), the car generation rate is much higher at 5.6 cars per time step.

We use a subset of the full 24-hour period because it includes hours during which there is little traffic demand (i.e. early hours) and also a heavy traffic jam towards the late afternoon, which causes all vehicles to be at a stand still and provides little useful comparison for any traffic light algorithms (as all vehicle delays will tend towards infinity).

The LuST scenario was chosen as a test scenario because it was both a reasonable representation of a real European city, including the generated traffic demand which the authors claim was calibrated using available statistics, and also because its scale means that the simulations can be run in a reasonable time frame and produce a manageable amount of data. Other scenarios such as TAPASCologne [105] were considered, but their immense scale requires considerably more time and computing power, and it was not clear what could be learned from purely increasing the size of the city being tested.

Table 7.1: Luxembourg Network Topology

Topology	**Junctions**	**Joining Edges**	**Traffic Lights**
Luxembourg	2112	5969	201

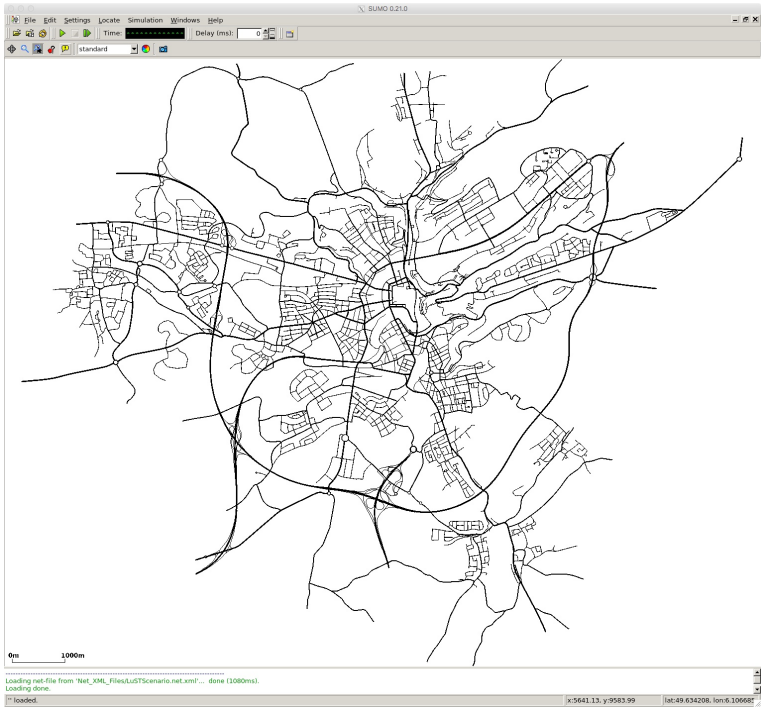


Figure 7.1: The Luxembourg network as visualised in the traffic simulator SUMO

Figure 7.2: Luxembourg Network

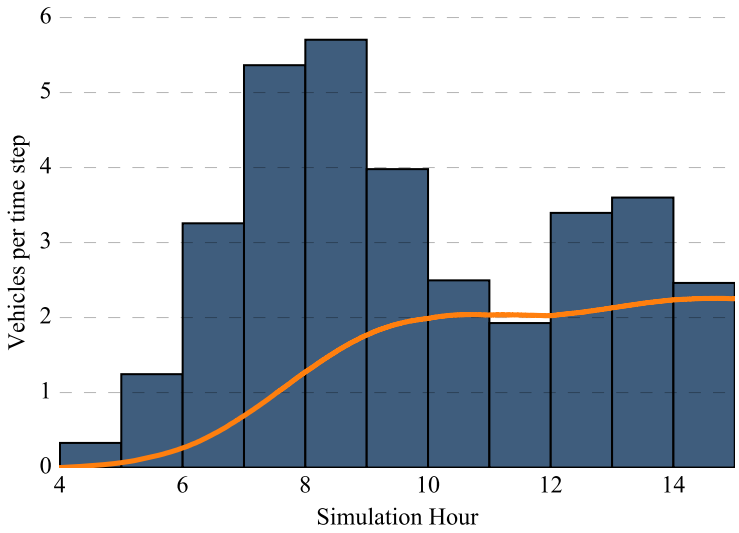


Figure 7.3: Traffic demand in the Luxembourg scenario. Blue bars show the mean car generation rate for each hour, whilst the orange line indicates the mean car generation rate throughout the 11-hour period. Morning rush-hour is visible at hour 8, and a lunch time rush-hour is visible at hour 13.

7.1.1 Simulation Set-Up

The scenario is given with two sets of routes, one determined via Dijkstra's algorithm and another with a Dynamic User Equilibrium (DUE) determined using Gawron's method [41]. The nature of our simulation will change the equilibrium in the network, and so we have not used the DUE routes. Instead, we have used a one-shot dynamic routing method, which does not give a stochastic user equilibrium, however it allows vehicles to plan at the start of their journey to avoid unnecessary congestion. We also use several SUMO processing options which are recommended for running the scenario, and these options are detailed in Table 7.2. These parameters allow for vehicles to be discarded when they wait longer than 10 minutes to begin their journey, or 'teleport' to the next road in their journey when they have yielded for longer than 10 minutes. These processing options are useful in complicated scenarios such as this, which can become unrealistically grid-locked due to a single stuck vehicle. We record statistics for the number of vehicles that do not start their journey or are teleported. These values are low and comparable across all simulations. The usefulness of teleportations during simulations and their possible impact on the results is discussed further in Section 7.4.

We find that for all simulations around 1.2% of vehicles do not start their journey due to a delayed start.

Table 7.2: Luxembourg Simulation SUMO Simulation Parameters

Parameter	Value
ignore-junction-blocker	20
time-to-teleport	600
max-depart-delay	600

7.2 Numerical Validation of Congestion-Aware and Capacity-Aware Stage Selection in the Luxembourg Scenario

Here we present results obtained for each simulation of 13-hours of traffic demand. We present results on the average flow through the network, mean trip duration, and mean wait time.

7.2.1 Mean Flow Through the Luxembourg Network

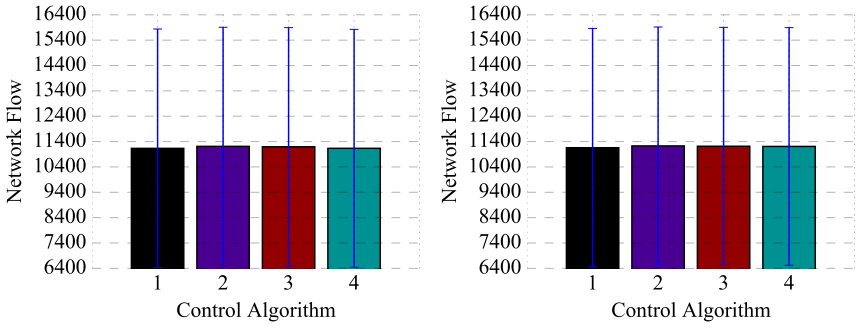
In Figure 7.4 we observe the mean flow in the network for the duration of the simulation. We find that the best performance is given by the congestion-aware and capacity-aware stage selection algorithms when combined with the Tmin/Tmax stage duration algorithm in scenarios B and D. Both give an average flow through the network of 11228 vehicles per hour. In comparison to the best performing fixed-cycle controller, this is a 0.6% increase in flow over the fixed interval scheme. The error bars show deviation in flow between hours, which is large because the flow through the network varies significantly over the 11-hour period.

7.2.2 Mean Trip Duration for All Vehicles

In Figure 7.5 we present the mean trip duration for all vehicles in the network. We find that where our intersection controller increased the flow in the network, it also reduced the mean travel time in the network. In absolute terms, we see a 1.4-minute reduction in the mean trip duration when comparing a 15-second fixed-cycle controller with the Tmin/Tmax stage duration algorithm in scenarios B and D. This is equivalent to a 15% reduction in the mean trip duration. The error bars show the deviation in mean trip duration for all trips taken over the 11-hour period. Due to the large number of vehicles, varied journeys distances and changing network conditions over the 11-hour period, we see that there is a large deviation in mean trip duration between vehicles (and again for mean wait time which we show in Figure 7.6), however, we reiterate that these results are preliminary investigations, which show the overall effect of the intersection controllers on mean trip duration.

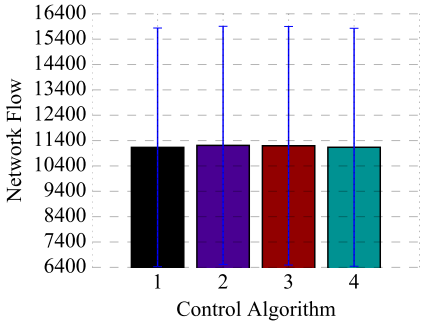
7.2.3 Mean Wait Time for All Vehicles

In Figure 7.6 we see the mean wait time in the network. We find that where our intersection controller reduced trip duration, there is usually a corresponding decrease in the wait

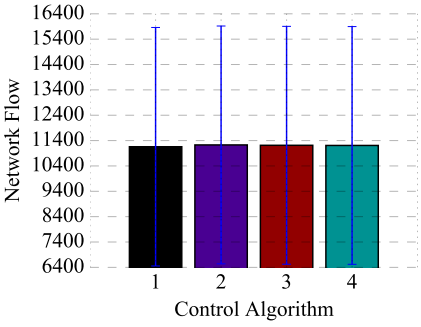


(a) Scenario A

(b) Scenario B

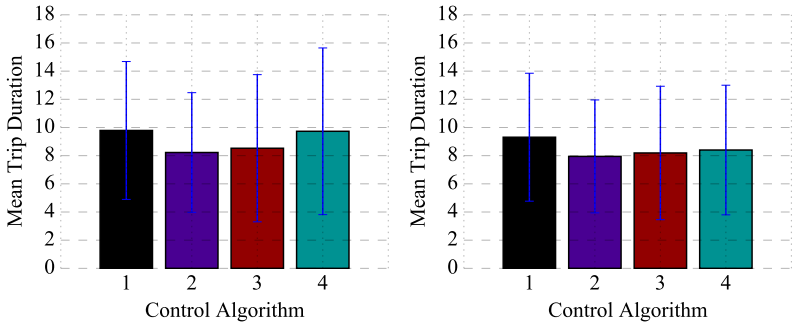


(c) Scenario C



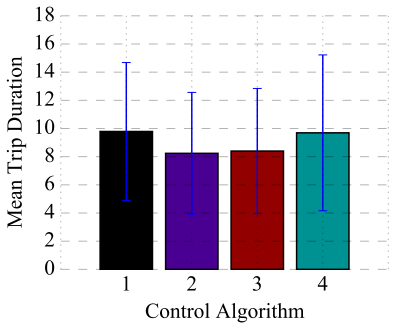
(d) Scenario D

Figure 7.4: Mean flow during the simulation for (1) Fixed-cycle (2) Tmin/Tmax (3) Proportional (4) Model based stage duration. Flow is measured in vehicles per hour. Note that the y-axis begins at 11000 vehicles per hour. The error bars show standard deviation over the 13-hour simulation, which reflects that some hours will have had much higher or lower flows than the mean.

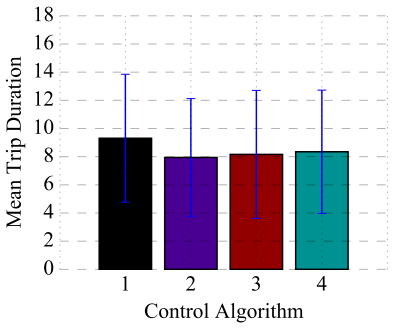


(a) Scenario A

(b) Scenario B

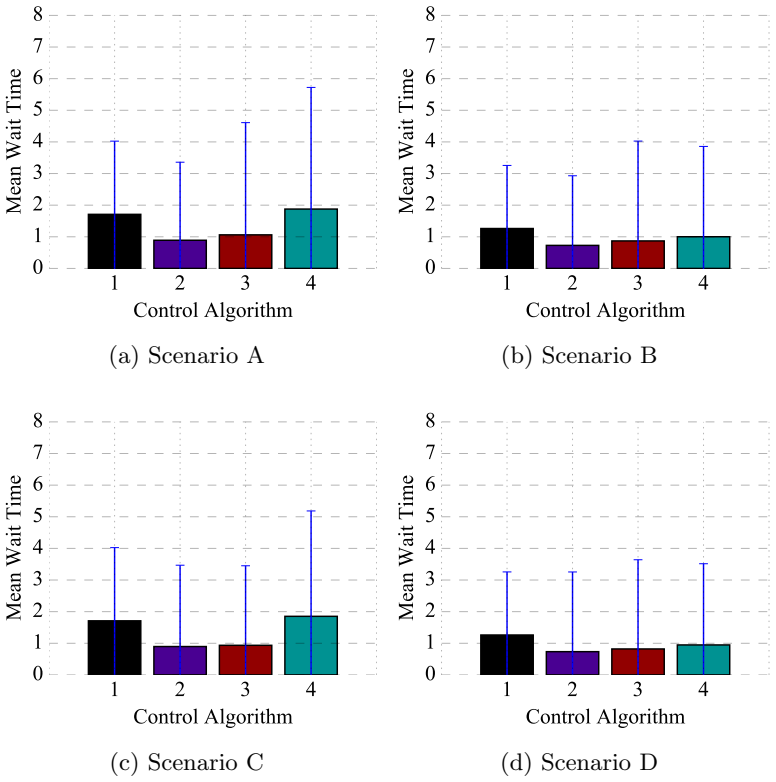


(c) Scenario C



(d) Scenario D

Figure 7.5: Mean trip duration during the simulation for (1) Fixed interval (2) Tmin/Tmax (3) Proportional (4) Model based stage duration. Time is measured in minutes. The error bars show standard deviation in trip duration for all vehicles. Given the size of the network and the vehicle loading this reflects that many vehicles experience longer journeys and hence much longer or shorter trip durations than the mean.



(a) Scenario A

(b) Scenario B

(c) Scenario C

(d) Scenario D

Figure 7.6: Mean wait time during the simulation for (1) Fixed interval (2) Tmin/Tmax (3) Proportional (4) Model based stage duration. Time is measured in minutes. The error bars show the standard deviation in wait time for all vehicles. This reflects that many vehicles experienced much longer wait times than the mean.

time, however, the absolute reduction in trip duration may be greater than the reduction in waiting time, indicating an overall improvement in road speed across the network. When comparing a 15-second fixed-cycle controller with the Tmin/Tmax stage duration algorithm in scenarios B and D, we see a reduction in wait time from 1.3 minutes to 0.7 minutes or a 46% reduction in waiting time. As with mean trip duration, the deviation shown by the error bars is caused by the large distribution of differing vehicles journeys across the network, but these results are intended to give a preliminary overview of the potential performance gains of our intersection controller.

7.3 Numerical Validation of the Propagating Pressure Controller in the Luxembourg Scenario

Simulations were run for the PPC, OBB and Gregoire algorithms in order to compare them with fixed-cycle control, and our other stage selection and stage duration algorithms. As the Luxembourg network is much larger than the synthetic networks tested in Chapter 5 it contains many intersections which do not have traffic light control, and these intersections halt the propagation of pressure in our PPC. For this reason, we tested two versions of the PPC on the Luxembourg network:

1. Only traffic light controlled intersections have an intersection controller. These intersections measure queues and propagate pressures as per the algorithm. If the next intersection downstream does not have an intersection controller (i.e. traffic lights), then propagation is halted.

2. All intersections have an intersection controller. These controllers measure queue lengths and propagate pressures through the network, however, only intersections which have traffic lights can use these measurements for stage selection and traffic light control.

In this section, we first compare the two versions of the PPC, before comparing all of the algorithms tested in the Luxembourg network against each other.

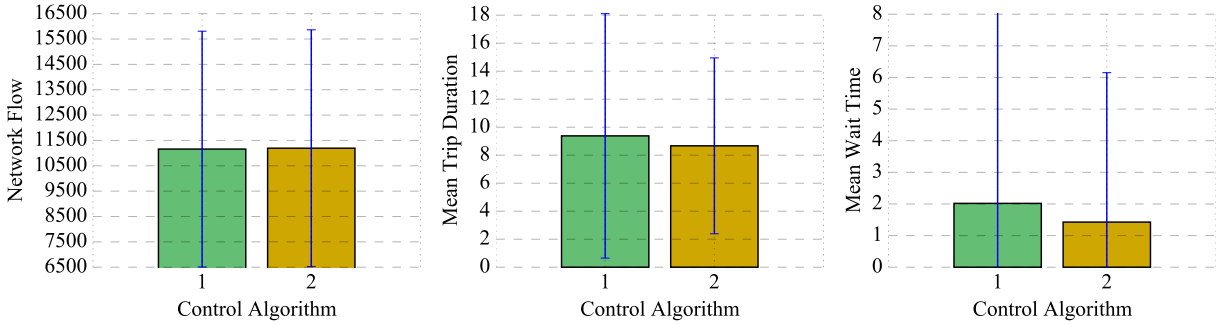
7.3.1 Comparing the Two Versions of the PPC

The results from simulations of the PPC are shown in Figure 7.7 and summarised in Table 7.3.

In Figure 7.7b we observe that the mean trip duration for all journeys is 9.4 minutes when only traffic light controlled intersections propagate pressures and that this is reduced to 8.7 minutes when all intersections can propagate the pressures.

In Figure 7.7c we find that the mean wait time is 2 minutes for the traffic light only version of the PPC stage selection algorithm, and reduces to 1.4 minutes when it is extended to all intersections.

Overall these results demonstrate that communication between all intersections is preferable for the performance of the PPC, rather than only at traffic light controlled intersections. However, even when all intersections have the ability to propagate queue lengths,



(a) Flow (b) Trip Duration (c) Wait Steps

Figure 7.7: Network flow, mean durationwait time during the simulation for (1) Occupancy Based Backpressure (2) Gregoire Capacity Aware (3) PPC Stage Selection (traffic light controlled intersections only) (4) PPC State Selection (at all intersections). Error bars reflect standard deviation from the mean. Due to this being a simulation of varied traffic loading over 13-hours with different routes and route lengths the error bars show that the flow from hour to hour, and the experience of many vehicles, differs significantly from the mean.

Table 7.3: Summary of controller performance in the Luxembourg scenario. Pressure Propagating Stage Selection and comparison stage selection algorithms.

Metric	PPC (traffic light only)	PPC (all intersections)
Mean Flow	11159	11194
Mean Trip Duration	9.4	8.7
Mean Wait Time	2.0	1.4

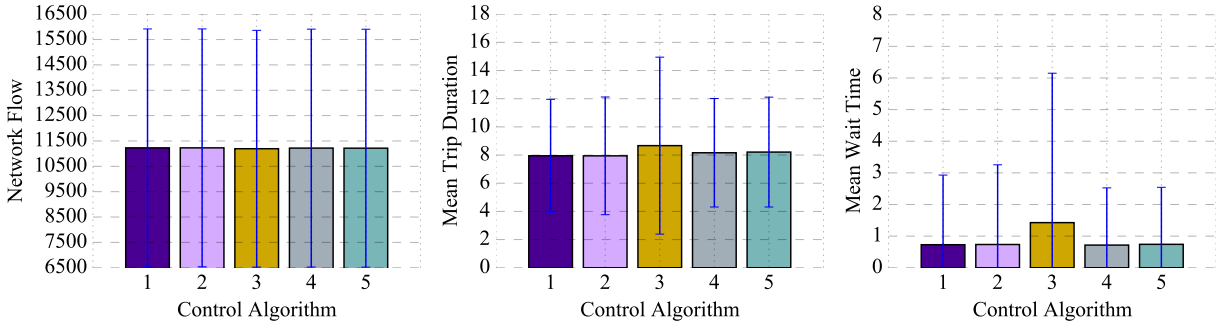
the results are not as promising as found with the Capacity-Aware and Congestion-Aware stage selection algorithms in combination with the Tmin/Tmax stage duration algorithm.

Table 7.4: Summary of controller performance in the Luxembourg scenario for Fixed-Cycle, Congestion-Aware and Capacity-Aware Stage Selection.

	Uncontrolled		**Controlled**											
	Fixed-Cycle		**Tmin/Tmax**				**Proportional**				**Model Based**			
Metric	15	31	A	B	C	D	A	B	C	D	A	B	C	D
Mean Flow	11157	11132	11211	11228	11211	11228	11194	11215	11196	11214	11133	11207	11136	11208
Mean Trip Duration	9.3	9.8	8.2	7.9	8.2	7.9	8.5	8.2	8.4	8.2	9.7	8.4	9.7	8.3
Mean Wait Time	1.3	1.7	0.9	0.7	0.9	0.7	1.1	0.9	0.9	0.8	1.9	1.0	1.9	0.9

Table 7.5: Summary of controller performance in the Luxembourg scenario for all algorithms.

Metric	Congestion-Aware Tmin/Tmax ($\tau_{\min} = 5$, $\tau_{\max} = 25$)	Capacity-Aware Tmin/Tmax ($\tau_{\min} = 5$, $\tau_{\max} = 25$)	PPC (all intersections)	Occupancy-Based Backpressure	Gregoire Capacity Aware
Mean Flow	11228	11228	11194	11218	11215
Mean Trip Duration	7.9	7.9	8.7	8.2	8.2
Mean Wait Time	0.7	0.7	1.4	0.7	0.7



(a) Flow (b) Trip Duration (c) Wait Steps

Figure 7.8: Network flow, mean durationwait time during the simulation for (1) Occupancy Based Backpressure (2) Gregoire Capacity Aware (3) PPC Stage Selection (traffic light controlled intersections only) (4) PPC State Selection (at all intersections). Error bars reflect standard deviation from the mean. Due to this being a simulation of varied traffic loading over 13-hours with different routes and route lengths the error bars show that the flow from hour to hour, and the experience of many vehicles, differs significantly from the mean.

7.3.2 Comparison of Best Performing Controller in the Luxembourg Network

Results comparing our congestion-aware and capacity-aware controllers, with the PPC, Gregoire and OBB algorithms is found in Figure 7.8. In these figures, the PPC has been implemented so that all intersections propagate pressures.

In Figure 7.8a we observe that the mean flow in the network is similar for all of the algorithms (see Table 7.5 for exact figures). In Figures 7.8b and 7.8c we observe that the PPC stage selection algorithm performs considerably less well than the other algorithms. We find that both the OBB and Gregoire algorithms achieve a mean trip duration of 8.2 minutes and a wait time of 0.7 minutes. This wait time is equal to that found for our congestion-aware and capacity-aware stage selection algorithms, however, we reduce the mean trip duration by 0.3 minutes using our algorithms.

7.4 Discussion of the Real World Scenario

In the Luxembourg scenario, we do not observe the same drastic improvement as in the synthetic scenarios over the best controlled and uncontrolled schemes. Furthermore, the PPC performs the least well of all the algorithms (in fact it is not notably better than even the fixed-cycle controller). We believe that this may be caused by the Luxembourg scenario operating in a region past the critical generation rate for all scenarios. Fine tuning of the loading may provide cases where the PPC outperforms the other controllers or at least outperforms the fixed-cycle.

We observe that whilst a reduction in trip duration correlates with a reduction in the waiting time, the reduction in duration can be often greater than the reduction in wait time. This indicates that the intersection controller also contributes to the total flow of vehicles along roads, presumably maintaining a greater average road speed throughout the

network.

It is notable that in Figure 7.8 there is no discernible difference in performance between congestion-aware and capacity-aware stage selection algorithms. We would have expected the more sophisticated Capacity-Aware stage selection to perform better, however, as we found in the synthetic networks there does not seem to be a significant advantage to using it. Given the much simpler implementation of Capacity-Aware stage selection, it seems that this would be the preferable algorithm to use.

7.4.1 Investigating the Final State of the Stage Duration Algorithms

In order to understand the variation between the stage duration algorithms, we show a histogram of the final green times for all of the intersections in the Luxembourg network in Figure 7.9. Results are shown for scenario D, where all 3 controllers outperformed the fixed interval control. We observe three distinct distributions of stage durations for each algorithm.

- In the Tmin/Tmax algorithm we see that the largest group of stage durations are at the minimum value of 5 seconds (around 36% of all stage durations). There is a fat-tailed distribution of stage durations up to the maximum value of 25 seconds. The second largest group is 10 seconds, which is equivalent to a stage being selected and reduced once (possibly to a local optimum).

- In the proportional algorithm, which is not bounded, we find that the distribution has a very long tail, with a minimum green time of 0 seconds, and a maximum green time of 93 seconds. Around 10% of the green times are 15 seconds, which may indicate that these stages were either never selected or were already at a local optimum at the end of the simulation.

- The model based algorithm has 96% of the stage durations set at 5 seconds, and the rest at 25 seconds.

Understanding the stage duration algorithms, and looking at the distributions of their green times, it appears that the performance may be related to how responsive the stage durations are to changes in the system.

- The model based stage duration is very responsive to changes in the system. Using constantly updated rates of inflow and outflow, combined with the current queue length, it selects the stage duration at the beginning of the stage. This means it will adapt faster to the current traffic conditions, but it may be too responsive and does not calculate an error in the green time as with the other two stage duration algorithms.

- The proportional and Tmin/Tmax stage duration algorithms have some hysteresis. They will only gradually change to match growing or shrinking queue lengths. As proportional stage durations are unbounded above 0 they can become very long, which may give poor performance if the traffic conditions change suddenly.

- The Tmin/Tmax stage duration is not as responsive as the model based (it will go from Tmin to 95% of Tmax in 6 stages), but will respond much faster than the proportional algorithm and is bounded. The Tmin/Tmax algorithm allows for stage durations to adjust to changing conditions, but slowly enough that small perturbations in the queue lengths don't make large changes to the green times.

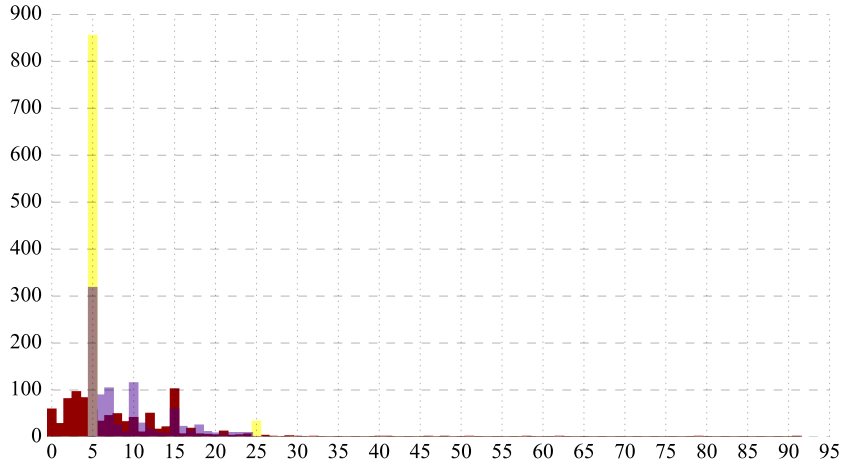


Figure 7.9: Histogram of green times for the three stage duration controllers in scenario D: Tmin/Tmax (purple), Proportional (maroon), and Model Based (blue).

7.4.2 The Magnitude and Effect of Teleportations on the Simulation

The effect of teleporting in SUMO, which happens when vehicles collide or are stuck in one place for too long, is discussed here, as it must be considered whether it can affect the realism of the simulations. Teleportation can be switched off but it is often necessary for stopping small simulation artefacts causing unrealistic gridlocks and bottlenecks. As teleportation may hide cases where vehicles yield for a long time due to the controller behaviour rather than simulation errors, we provide details of teleportations for comparison. Table 7.6 summarises these statistics. We find that the number of teleportations per vehicle ranges between 0.004 and 0.007, or in other words between 4 and 7 of every 1000 vehicles teleported once during their trip. The percentage of trips which were cancelled ranges between 0.83% and 1.46%, or in real terms between 1341 and 2359 vehicles cancelled their journeys due to waiting too long.

We find that in general, the number of teleportations is lower for better-performing algorithms, which indicates that the algorithm is performing well, rather than benefiting from unrealistic simulation behaviour. However, there are some discrepancies. Occupancy Based Backpressure and Gregoire Capacity-Aware have the lowest number of teleportations (4 and 4.2 per 1000 vehicles respectively) but did not perform as well as the Capacity-Aware and Congestion-Aware stage selection when combined with Tmin/Tmax stage duration (4.7 and 4.5 teleportations per 1000 vehicles). Occupancy Base Backpressure also had a lower percentage of cancelled trips (0.83% compares with 0.9 and 1.1%).

The question this raises is whether a difference of 82 additional teleportations (0.7 per 1000 vehicles) and 111 additional vehicle trips (0.07% of the total trips) is enough to yield a significant difference in the mean trip duration. During the Congestion-Aware & Tmin/Tmax simulation 159558 vehicles completed journeys. If we assume that the actual travel time for all of the vehicles less the 82 extra that teleported was 8.2 minutes and that the teleportations reduced the travel time of the affected vehicles to 0, then we find that the average travel time would still be 8.196 minutes. We can, therefore, assume that this is not responsible for the algorithm's performance. Similarly in order for the difference in travel to to have occurred due to the additional vehicle trips, either the 111 vehicles must have added a total of 800 hours to the total travel time for all vehicles, or their own journeys must have taken 439 minutes on average (in which case they would have teleported due to waiting). It therefore, seems unlikely that the improved performance of our algorithm is due to an artefact of the simulation.

It is notable that the algorithms with the longest mean trip duration (using Model Based stage duration algorithms) had the lowest number of emergency stops. These kinds of secondary considerations are overlooked when considering the efficiency of the network, and consideration should be given to whether algorithms which can lead to short stage durations might lead to accidents if implemented in a real-world network.

Table 7.6: Statistics for the number of emergency stops, teleports and cancelled trips in the simulations. Simulations are ordered by mean trip duration (lowest to highest)

Simulation	Emergency Stops	Teleports	Teleports per 1000 vehicles	Cancelled Trips	% Cancelled Trips
Congestion-Aware & Tmin/Tmax ($\tau_{\min} = 5, \tau_{\max} = 25$)	766	752	4.7	1452	0.9
Capacity-Aware & Tmin/Tmax ($\tau_{\min} = 5, \tau_{\max} = 25$)	802	711	4.5	1777	1.1
Occupancy Based Backpressure (15s Fixed Interval)	811	670	4.2	1341	0.83
Gregoire Capacity-Aware (15s Fixed Interval)	787	632	4	1547	0.96
Capacity-Aware & Proportional ($K_p = 0.15, \tau_0 = 15$)	778	812	5.1	2245	1.39
Congestion-Aware & Tmin/Tmax ($\tau_{\min} = 5, \tau_{\max} = 55$)	666	801	5	2070	1.28
Capacity-Aware & Tmin/Tmax ($\tau_{\min} = 5, \tau_{\max} = 55$)	701	788	5	1838	1.14
Congestion-Aware & Proportional ($K_p = 0.15, \tau_0 = 15$)	735	771	4.9	2236	1.39
Capacity-Aware & Model Based ($\tau_{\min} = 5, \tau_{\max} = 25$)	588	786	4.9	1765	1.1
Capacity-Aware & Proportional ($K_p = 0.15, \tau_0 = 30$)	693	794	5	2359	1.46
Congestion-Aware & Model Based ($\tau_{\min} = 5, \tau_{\max} = 25$)	549	860	5.4	1603	0.99
Congestion-Aware & Proportional ($K_p = 0.15, \tau_0 = 30$)	625	931	5.9	2328	1.44
Propagating Pressure (15s Fixed Interval)	683	1107	7	1803	1.12
Congestion-Aware & Model Based ($\tau_{\min} = 5, \tau_{\max} = 55$)	312	921	5.8	2093	1.3
Capacity-Aware & Model Based ($\tau_{\min} = 5, \tau_{\max} = 55$)	367	949	6	1553	0.96

Chapter 8

Conclusions and Future Work

The purpose of this thesis was to explore control strategies for reducing congestion in road networks, expand them in the direction of decentralised strategies, and present some analysis of the effect of topology and scale on the performance of these algorithms. We have presented several such algorithms in routing and intersection control and explored various network topologies with each.

In the first instance, we proposed that a simple vehicle routing algorithm, which encouraged vehicles to take an alternative route if the occupancy of the next road on the shortest path was above a threshold value. The aim of the algorithm was to prevent the onset of congestion as vehicles would perform load balancing collaboratively, and that by finding the correct value for a tuning parameter would balance an individual vehicle's to take the shortest route, against the desire of all vehicles to avoid a congested network with large delays.

The results showed that network topology had a large impact on the capacity of the network, even when the number of edges in the network was constant. In the lattice and spiderweb networks, we found that by taking alternative paths based only on the occupancy of the next road we gained a drastic improvement over routing using only the shortest path. The algorithm is not sophisticated and extremely easy to implement, but the load balancing was pronounced across the network for vehicles only using shortest travel time calculations to find their route (there were no dynamic travel times available to the vehicles). However, when applied to networks with random or small-world degree distributions we found that the performance dropped, especially in comparison to a Dynamic User Equilibrium (the distribution we expect when users are allowed to pick their own routes).

We presented an analysis of the finding that in some networks a clearly optimal value of the tuning parameter α could be identified, and what this indicated about the behaviour of the algorithm. We found that we could demonstrate logical reasons behind the optimal value for each network, and the behaviour this would elicit in the vehicles.

The findings of Chapter 4 indicate that a simple load balancing by individual vehicles is sufficient in some cases to prevent gridlock in the network. The finding is significant as methods in the literature generally consider that only a central controller is able to drive traffic towards a more optimal state. As more data becomes available there will

undoubtedly be more significant indicators than occupancy which could be used to balance against estimated travel times in the network.

In our chapters on intersection control, we explored a decentralised algorithm based on the principle of work-conservation, which had been shown to be effective in the literature. We developed our own modular control algorithms based on this principle and tested them on both synthetic networks and a real-world example of Luxembourg. Our algorithms demonstrate excellent performance compared to fixed-cycles with very little tuning. Interestingly we also found that different algorithms had qualitatively different properties, with some giving shorter minimum trip durations, but other algorithms able to provide a higher maximum flow through the network.

We also found that in our real-world scenario the most sophisticated algorithm (Propagating Pressure Control) performed the least well of our selected algorithms. This ran counter-intuitive to our expectations from results in the synthetic networks and highlighted the issue of translating results from synthetic networks into reality. However, our results were not completely surprising, in that an algorithm which had been promising in the lattice and random networks gave the best performance in the Luxembourg network.

Working on the ideas in this thesis has led to many ideas and scenarios which we were not able to test or develop due to time and resources. There has also been a huge amount of development over the last 4 years in the news and research surrounding connected and driverless vehicles which have given some more context to this area of study.

In the realm of vehicle routing, there are a huge number of cost functions which could be incorporated into a decentralised routing algorithm. An algorithm in the real-world might make it's way to drivers through a service which delivers real-time traffic information such as Google Maps [87]. Instead of drivers tending towards a user equilibrium (based on taking the fastest travel time each time), then additional cost functions (such as occupancy) can be incorporated into the routing decision.

It would also be desirable to explore which observable metrics in the road network have the highest impact on reducing congestion. A genetic algorithm or similar could be used to explore a combination of numerous metrics.

In the context of intersection control, we would like to explore the impact of intersection control on route choice, as it has been suggested that traffic signals may have the effect of destabilising route choice models in a way not represented in our models. It would be interesting to study if a controller such as the PPC had the effect of causing drivers to switch to routes of "higher-pressure" if they were serviced more frequently by traffic lights.

A further area which we would like to develop is the comparison of synthetic scenarios with the real-world scenarios. The synthetic networks, whilst contrived in some respects, produce results that are easily comparable. The topologies are clearly defined and the loading on the network is homogeneous. The Luxembourg scenario in Chapter 7, whilst realistic in the sense of being a real road network, and loading that could be considered realistic, presents more challenges. We also found that in such a large network with such a broad range of trip lengths and network conditions there is a significant deviation between vehicle trips and the data is much harder to interpret. The results are also very specific

to the network in question.

A comparison with a centralised traffic-responsive strategy such as SCOOT in both the synthetic and Luxembourg networks would be desirable to determine how close our decentralised strategies are able to come to matching a centralised one, however the implementation of this is beyond the scope of this thesis due to both funding and time constraints.

In summary, this thesis has delivered the algorithms and numerical analysis it promised in the hope of finding control strategies which are more effective than those currently in use, while also being feasible to implement with current technology.

Abbreviations

BPR Beareau of Public Roads.

CSO Constrained System Optimal.

CTM Cell Transmission Model.

DRA Decentralised Routing Algorithm.

DSO Dynamic System Optimal .

DTA Dynamic Traffic Assignment .

DUA Dynamic User Assignment .

DUE Dynamic User Equilibrium .

GPS Global Positioning System.

ITS Intelligent Transportation Systems.

LTTR Live Travel Time Routing.

LWR Lighthill, Whitham and Richards.

O-D Origin-Destination .

OBB Occupancy Based Backpressure.

PPC Pressure Propagating Controller.

SO System Optimal .

SSO Static System Optimal .

SUE Static User Equilibrium .

TSP Travelling Salesman Problem.

UE User Equilibrium .

V2I Vehicle-to-Infrastructure .

V2V Vehicle-to-Vehicle.

VRP Vehicle Routing Problem.

Glossary

Congestion The antithesis of Free-Flow.

Dynamic System Optimal The System Optimal (SO) when the network is time-varying.

Dynamic User Equilibrium The User Equilibrium (UE) when the network is time-varying.

Dynamic User Assignment See Dynamic Traffic Assignment (DTA).

Dynamic Traffic Assignment Allocation of routes to vehicles in time-varying networks in order to reflect driver route choice patterns (i.e. for simulation), or to attempt to best route traffic.

Free-Flow Traffic is able to flow without obstruction. The flow along a road will increase linearly with an increase in traffic density or velocity.

Macroscopic Model of road networks based on flows along road and in and out of intersections.

Microscopic Model of road networks based on calculation of individual movements of vehicles.

Static System Optimal The SO when the network is not time-varying.

Static User Equilibrium The UE when the network is not time-varying.

System Optimal The optimal distribution of vehicles to minimise total travel time in the network.

User Equilibrium The equilibrium reached when no vehicles can change their route for a faster journey.

Vehicle to Infrastructure Communication Communications protocol enabling vehicle communication with road infrastructure (e.g. traffic lights).

Appendices

Appendix A

Futher Details of Optimal Control Models from Section 2.3.2

A.1 Papagerogiou Model of Network Nodes

A static model of network nodes can be constructed, which describes the splitting of traffic at each intersection towards destination nodes.

The total flow q_{nj} leaving node n and destined for node j is the sum of the flows destined for node j flowing on each of the input links m to node n, additionally any traffic flow originating from node n for node j must be included. Mathematically this is expressed as,

$$q_{nj} = \sum_{m \in I_n} Q_m \Gamma_{mj} + d_{nj}, \quad n \in N,\ j \in S^n \tag{A.1}$$

where I_n is the set of links entering node n, Q_m is the traffic volume leaving link m, Γ_{mj} is the proportion of flow exiting link m which is destined for node j (output composition rate), and d_{nj} is the traffic demand from node n to node j.

The splitting rate, β_{nj}^m, is the proportion of traffic travelling along road m from node n to node j,

$$\beta_{nj}^m = \frac{q_{nj}^m}{q_{nj}} \tag{A.2}$$

The input composition rate, γ_{mj}, is the proportion of flow into link m destined for node j, such that,

$$\gamma_{mj} = \frac{q_{nj}^m}{q_m} = \beta_{nj}^m \cdot \frac{q_{nj}}{q_m} \tag{A.3}$$

A.2 Papageorgiou Model of Network Links

Network links are dynamic models which describe the transformation of input flows and composition rates (q_m, γ_{mj}) from the node models into output flows and composition rates

of the links (Q_m, Γ_{mj}).

The state space form can broadly be expressed as,

$$\mathbf{Y}(k) = \mathbf{G}[\mathbf{x}(k), \mathbf{U}(k)] \tag{A.4}$$

$$\mathbf{x}(k+1) = \mathbf{F}[\mathbf{x}(k), \mathbf{U}(k)] \tag{A.5}$$

Where the output vector, $\mathbf{Y}$, comprises the output flows and composition rates, and the input vector , $\mathbf{U}$, comprises the input flows and composition rates. The state vector, $\mathbf{x}$, is model dependent.

Physical constraints such as capacity can be expressed as inequalities,

$$\mathbf{H}[\mathbf{x}(k), \mathbf{U}(k)] \leq 0 \tag{A.6}$$

A.3 Kachroo Link-Based Model

The link based model closely follows work laid out by Papageorgiou. The conservation of flow gives the following formulation for the change of traffic density over time, $\dot{\rho}_m$, which is related to the difference between input flow, q_m, and output flow, Q_m. The output flow in turn is related to traffic density, ρ_m (and road specific constants such as the jam density, κ_m, and the maximum inflow, $q_{\text{max,m}}$), which is common in many models in the literature.

$$\dot{\rho}_m(t) = \frac{1}{\Delta_m}(q_m(t) - Q_m(t)) \tag{A.7}$$

$$Q_m(t) = q_{\text{max,m}}(1 - \exp^{\frac{\rho_m}{\kappa_m}}) \tag{A.8}$$

The composition rates are then expressed using either (i) a time delay model s.t. $\Gamma_{mj}(t) = \gamma_{mj}(t - \tau)$, where delay, τ is related to link travel time or (ii) a first order filter model s.t. $\dot{\Gamma}_{mj}(t) = \frac{v_m}{\Delta_m}(\gamma_{mj} - \Gamma_{mj})$, where $v_m = \frac{Q_m}{\rho_m}$. Kachroo recommends use of the filter model; the delay model creates an infinite dimensional system, and the filter model reflects some propagation of the composition rate along the link, which is seen as representative of real system behaviour.

The measurement function is given by assuming that travel time on a link is given by,

$$h(x) = \sum_{i \in M} \frac{\Delta_i \cdot \rho_i}{Q_i}, \tag{A.9}$$

where M is the set of all links in the network.

A.4 Kachroo Route-Based Model

Kachroos route-based model uses changes in destination based densities to directly calculate composition rates, such that,

$$\dot{\rho}_{mj}(t) = \frac{1}{\Delta_m}(\gamma_{mj}(t) \cdot q_m(t) - \Gamma_{mj}(t) \cdot Q_m(t)), \tag{A.10}$$

where,

$$\Gamma_{mj}(t) = \frac{\rho_{mj}(t)}{\rho_m(t)} \tag{A.11}$$

The form of the measurement function is given as,

$$h(x) = \frac{\left(\sum_{i \in M} \Delta_i\right) \cdot \left(\sum_{j \in S_i} \rho_{ij}\right)}{Q_i}, \tag{A.12}$$

where M is the set of all links in the network, and S_i is the set of destination nodes reachable via link i.

Appendix B

Notation

B.1 Routing Notation Reference (Chapter 4)

B.1.1 Network Topology

Symbol	Description	Definition
$\mathbf{Q}$	a bounded region	$\mathbf{Q} \in \mathbb{R}^2$
$\mathbf{G}$	the graph representing the road network	$\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$
$\mathbf{V}$	the set of m vertices representing junctions	$\mathbf{V} = \{1, 2, ..., m\}$
$\mathbf{E}$	the set of edges representing the roads connecting junctions	-
v_j	the j-th junction	$v_j, \quad j \in \mathbf{V}$
(v_j, v_k)	an arbitrary road between two junctions v_j and v_k	$(v_j, v_k) \in \mathbf{E}$

B.1.2 Vehicle Properties

Symbol	Description	Definition
t	a continuous time step	-
N	number of vehicles	-
$\mathbf{I}$	the set of all vehicles	$\mathbf{I} = \{1, 2, ..., N\}$
$\mathbf{s}_i$	starting position of the i-th vehicle	$\mathbf{s}_i \in \mathbf{Q},\ i \in \mathbf{I}$
$\mathbf{p}_i(t)$	the position of the i-th vehicle at time step t	$\mathbf{p}_i(t) \in \mathbf{Q},\ i \in \mathbf{I}$
$\mathbf{d}_i$	the destination of the i-th vehicle	$\mathbf{d}_i \in \mathbf{Q},\ i \in \mathbf{I}$

B.1.3 General Routing Algorithm Description

Symbol	Description	Definition
$J_i^{(j,k)}$	utility function for the i-th vehicle to take the road $(v_j, v_k) \in \mathbf{E}_j$	$J_i^{(j,k)} = \underline{\boldsymbol{\alpha}}^T \cdot \boldsymbol{\Phi}_i^{(j,k)}$
$\boldsymbol{\Phi}_i^{(j,k)}$	a vector of cost functions associated with the i-th vehicle taking road (v_j, v_k)	-
$\underline{\boldsymbol{\alpha}}^T$	a vector of coefficients associated with each cost function	-

The road taken is given by,

$$\min_{(v_j, v_k) \in \mathbf{E}_j} J_i^{(j,k)} \tag{B.1}$$

B.1.4 Metrics

Symbol	Description	Definition
$\mathcal{D}(v_j, v_l)$	the shortest travel time between two vertices in the network	-
$\mathcal{D}_i$	minimum expected travel time for the i-th vehicle	$\mathcal{D}_i = \mathcal{D}(\mathbf{s}_i, \mathbf{d}_i)$
$\mathcal{T}_i$	the actual travel time for the i-th vehicle	$\mathcal{T}_i = \mathcal{T}(\mathbf{s}_i, \mathbf{d}_i)$
ω_i	delay experienced by the i-th vehicle	$\omega_i = \mathcal{T}_i - \mathcal{D}_i$
$\bar{\mathcal{D}}i$	the mean expected travel time for all vehicles	$\bar{\mathcal{D}} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathcal{D}_i$
$\bar{\mathcal{T}}$	the mean travel time measured for all vehicles	$\bar{\mathcal{T}} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathcal{T}_i$
$\bar{\omega}(\lambda)$	the mean delay experienced by all vehicles at vehicle generation rate λ	$\bar{\omega} = \bar{\mathcal{T}} - \bar{\mathcal{D}}$
λ	the vehicle generation rate (number of vehicles entering the network at each time step)	-
$\hat{\omega}$	the acceptable delay threshold	$\hat{\omega} = \beta \cdot \bar{\mathcal{D}}$
β	the acceptable ratio between minimum expected travel time for a vehicles journey, and the actual delay it experienced	$\beta \in \mathbb{R}^+, \quad \beta \geq 0$
$\hat{\lambda}$	maximum car generation rate before delays go over the acceptable threshold	$\hat{\lambda} = \max\{\lambda \ : \bar{\omega}(\lambda) \leq \hat{\omega}\}$

We denote the number of cars entering the network at every time step as λ, and consider that the mean delay is a function of this, i.e. $\bar{\omega} = \bar{\omega}(\lambda)$. We further define delay as acceptable if it is less than some delay threshold, termed $\hat{\omega}$, where,

$$\hat{\omega} = \beta \cdot \bar{\mathcal{D}}, \quad \beta \in \mathbb{R}^+, \quad \beta \geq 0 \tag{B.2}$$

and note that $\bar{\omega} \leq \hat{\omega}$ implies that,

$$\bar{\mathcal{T}} \leq (1 + \beta)\bar{\mathcal{D}}, \quad \beta \geq 0 \tag{B.3}$$

where β is the acceptable ratio between the minimum expected travel time for a vehicles journey, and the actual delay it experienced.

B.1.5 Road Properties

Symbol	Description	Definition
$L^{(j,k)}(t)$	the number of vehicles currently using road (v_j, v_k)	-
$C^{(j,k)}$	the maximum number of vehicles that can fit onto road (v_j, v_k)	-
$\eta^{(j,k)}(t)$	the percentage of space on road (v_j, v_k) occupied by vehicles	$\eta^{(j,k)}(t) = \frac{L^{(j,k)}(t)}{C^{(j,k)}}$

B.1.6 Chosen Cost Functions

Symbol	Description	Definition
$\phi_i^{(j,k)}$	the travel time based cost function	$\phi_i^{(j,k)} = \phi(d_i, v_j, v_k)$ $0 \leq \phi_i^{(j,k)} \leq 1$
$\rho^{(j,k)}$	the occupancy based cost function	$\rho^{(j,k)} = \rho(\eta^{(j,k)}(t))$ $0 \leq \rho^{(j,k)} \leq 1$
α	the tuning parameter in our routing algorithm	$\alpha \in (0, 1]$

We calculate $J_i^{(j,k)}$ s.t.,

$$J_i^{(j,k)} = \begin{bmatrix} \alpha & (1-\alpha) \end{bmatrix} \begin{bmatrix} \phi_i^{(j,k)} \\ \rho^{(j,k)} \end{bmatrix} \tag{B.4}$$

Note that by tuning the control parameter α we can make the vehicle routing choice more or less sensitive to distance or congestion respectively.

B.2 Intersection Control Notation Reference (Chapter 5)

B.2.1 Intersection Topology

Symbol	Description	Definition
$\mathcal{V}$	the set of intersections in the network	-
i	the i-th intersection	$i \in \mathcal{V}$
I_i	the set of input links of the i-th intersection	-
l	a single input link	$l \in I_i$
O_i	the set of output links of the i-th intersection	-
m	a single output link	$m \in O_i$
j	a *phase* - any allowable movement from an input link l to an output link m (this may be referred to as the j-th phase or j-th queue)	$j = \{l, m\}$
n_i	number of phases at the i-th intersection	-
σ_i	the set of all phases at the i-th intersection	$\sigma_i = \{j_1, ..., j_{n_i}\}$

B.2.2 Queuing Process at Time Step k

Symbol	Description	Definition
k	a discrete time step	$k = 0, 1, ...$
$x_i^j(k)$	number of vehicles queueing for the j-th phase at the i-th intersection	-
$x_i^l(k)$	number of vehicles queueing on the l-th input link at the i-th intersection	-
$x_i^m(k)$	number of vehicles queueing on the m-th output link at the i-th intersection	-
$\lambda_i^j(k)$	arrivals to the j-th phase at the i-th intersection	-
μ_i^j	the saturation rate at which vehicles can depart from the queue (a constant)	-
$g_i^j(k)$	the status of the traffic light for the j-th phase at the i-th intersection (1 indicates green, 0 indicates red)	$g_i^j(k) \in [0, 1]$
c_i^m	capacity of the m-th output link a the i-th intersection (considered a physical property of the link)	-
ϵ_j	function relating the number of vehicles in the m-th output link to the capacity of the m-th output link, where $m \in j$	$\epsilon_j = \epsilon_j(x_i^m(k), c_i^m)$ (see (B.5))
$s_i^j(k)$	the service rate for the j-th phase at the i-th intersection	(see (B.6))

$\epsilon_j(x_i^m(k), c_i^m)$ is defined s.t.,

$$\epsilon_j(x_i^m(k), c_i^m) = \begin{cases} 1 & \text{if } x_i^m(k) < c_i^m \\ 0 & \text{otherwise} \end{cases} \tag{B.5}$$

The service rate is defined s.t.,

$$s_i^j(k) = g_i^j(k) \cdot \epsilon_j(x_i^m(k), c_i^m) \cdot \mu_i^j \tag{B.6}$$

Symbol	Description	Definition
$\mathbf{x}_i(k)$	State of the intersection (queue size for each phase) i.e. a $n_i \times 1$ column vector which is the the stack of all $x_i^j(k)$	-
P_i	the $z_i \times n_i$ stage matrix which is the stack of all stages in $\mathcal{P}_i$	-
$\mathbf{p}_i^q$	is the transpose of the q-th row of P_i, which is an $n_i \times 1$ column vector	-
$p_i^{q,j}$	refers to element the j-th row of p_i^q, which is a the value of g_i^j for the q-th stage	-
$\mathbf{u}_i(k)$	is the control input and is an $n_i \times 1$ column vector	-
$\mathbf{a}_i(k)$	is the $n_i \times 1$ column vector of arrival rates $(\lambda_i^j(k))$ at the i-th intersection	-
$D_i(k)$	is the $n_i \times n_i$ diagonal matrix of saturation rates (μ_i^j) at the i-th intersection	-
$E_i(k)$	is the $n_i \times n_i$ diagonal matrix of results for the function $\epsilon_j(x_i^m(k), c_i^m)$, where $m \in j$, for all $j \in \sigma_i$	-
$\mathbf{s}_i(k)$	is the $n_i \times 1$ column vector of service rates $(s_i^j(k))$ at the i-th intersection	(see (B.7))

B.2.3 General Stage Selection Algorithm Notation

Symbol	Description	Definition
q	a *stage* - a combination of g_i^j values for all phases this may be referred to as the q-th stage	$q = \{g_i^1, ..., g_i^{n_i}\}$
$\mathcal{P}_i$	the set of all possible stages for the i-th intersection	-
z_i	the number of stages in $\mathcal{P}_i$	-
$\mathbf{u}_i$	the control input derived from the set $\mathcal{P}_i$	$\mathbf{u}_i \in \mathcal{P}_i$

B.2.4 Compact Form Notation

The compact form of the service rate is given s.t.,

$$\mathbf{s}_i(k) = E_i(k) \cdot (D_i \cdot \mathbf{u}_i(k)) \tag{B.7}$$

The state space therefore becomes

$$\mathbf{x}_i(k+1) = [\mathbf{x}_i(k) + \mathbf{a}_i(k)_i(k) - \mathbf{s}_i(k)]_+ \tag{B.8}$$

For example, the state space model of a junction with two queues can be expressed as,

$$\begin{bmatrix} x_i^{j_1}(k+1) \\ x_i^{j_2}(k+1) \end{bmatrix} = \begin{bmatrix} x_i^{j_1}(k) \\ x_i^{j_2}(k) \end{bmatrix} + \begin{bmatrix} \lambda_i^{j_1} \\ \lambda_i^{j_2} \end{bmatrix} - \begin{bmatrix} \epsilon_{j_1} & 0 \\ 0 & \epsilon_{j_2} \end{bmatrix} \cdot \left(\begin{bmatrix} \mu_i^{j_1} & 0 \\ 0 & \mu_i^{j_2} \end{bmatrix} \cdot \begin{bmatrix} u_i^{j_1} \\ u_i^{j_2} \end{bmatrix} \right) \tag{B.9}$$

B.2.5 Capacity-Aware Stage Selection Algorithm

Symbol	Description	Definition
$\tilde{L}_i^q(\mathbf{p}_i^q)$	$n_i \times n_i$ matrix indicating if two phases share the same output link m during stage $\mathbf{p}_i^q$	-
$\tilde{\iota}_i^{q,j}$	sum of the elements on the j-th row of $\tilde{L}_i^q(\mathbf{p}_i^q)$	-
$\tilde{\mathbf{x}}_i^q$	Combined queue lengths	$\tilde{\mathbf{x}}_i^q = \tilde{L}_i^q(\mathbf{p}_i^q) \cdot \mathbf{x}_i$
$\tilde{x}_i^{q,j}$	j-th element of $\tilde{\mathbf{x}}_i^q$ relating to the j-th phase	-
υ	function relating queue length for the j-th phase and the capacity of the outgoing link m, where $m \in j$	$\upsilon = \upsilon(\tilde{x}_i^{q,j}(k), c_i^m)$ (see (B.10))
$\Upsilon_i^q(\tilde{\mathbf{x}}_i^q)$	the $n_i \times 1$ column vector which is the stack of the result of $\upsilon(\tilde{x}_i^{q,j}(k), c_i^m)$ for all n_i phases	-
$J_i^q(\mathbf{p}_i^q)$	utility function (for the q-th stage)	$J_i^q(\mathbf{p}_i^q) = \mathbf{p}_i^{qT} \cdot \Upsilon_i^q(\tilde{\mathbf{x}}_i^q)$

$\upsilon(\tilde{x}_i^{q,j}(k), c_i^m)$ is defined s.t.,

$$\upsilon(\tilde{x}_i^{q,j}(k), c_i^m) = \begin{cases} \frac{\tilde{x}_i^{q,j}(k)}{\tilde{\iota}_i^{q,j}} & \text{if } \tilde{x}_i^{q,j}(k) < (c_i^m - x_i^m(k)) \\ \frac{(c_i^m - x_i^m(k))}{\tilde{\iota}_i^{q,j}} & \text{otherwise} \end{cases} \tag{B.10}$$

where $m \in j$

The control input for stage selection is found by maximising the utility function J_i^q,

$$\mathbf{u}_i = \max_{\mathbf{p}_i^q \in \mathcal{P}_i} J_i^q(\mathbf{p}_i^q) \tag{B.11}$$

B.2.6 Pressure Propagating Controller (PPC) Stage Selection Algorithm

Symbol	Description	Definition
$\mathcal{B}_i^l$	the set of all vehicles on input link l at the i-th intersection	-
$\mathcal{B}_i^j$	the set of all vehicles assigned to phase j at the i-th intersection	$\mathcal{B}_i^j \subset \mathcal{B}_i^l \iff l \in j$
b	the b-th vehicle in a queue, which has an input link and an output link	$b = \{l, m\}$
$y_l^b(k)$	the queue position of the b-th vehicle on input link l i.e. $y_l^b = 1$ for the vehicle at the front of input link l	-
$\bar{x}_i^j$	the position weighted queue length of the j-th phase at the i-th intersection	$\bar{x}_i^j = \sum_{b \in \mathcal{B}_i^j} \frac{1}{y_l^b}$
$b_l^*(k)$	vehicle at the front of of the queue on input link l i.e. $y_l^{b_l^*} = 1$	-
$\bar{r}_i^l(k)$	pressure propagated from upstream intersections into input link l	-
$\hat{x}_i^j(k)$	the queue length for the j-th phase, used to find the optimal stage at each intersection	$\hat{x}_i^j(k) = \bar{x}_i^j(k) + \bar{r}_i^l(k), \quad l \in j$
$\hat{\mathbf{x}}_i(k)$	the $n_i \times 1$ column vector which is the stack of all $\hat{x}_i^j(k)$	-
$r_i^{l,m}(k)$	pressure propagated to from input link l to output link m pressure always propagates in the direction of travel of the vehicle at the front of the queue on input link l	$r_i^{l,m}(k) = \bar{r}_i^l(k) + x_i^l(k), \quad m \in b_l^*(k)$
$J_i^q(\mathbf{p}_i^q)$	utility function (for the q-th stage)	$J_i^q(\mathbf{p}_i^q) = \hat{\mathbf{x}}_i^T(k) \cdot E_i(k) \cdot \mathbf{p}_i^q$

B.2.7 General Stage Duration Algorithm Notation

Symbol	Description	Definition
$\tau_i^q(k)$	the stage duration of the q-th stage	-
$\bar{\eta}_i$	target fraction of vehicles to remove from i-th intersection during q-th stage	-
$\delta_i^q(k)$	target number of vehicles to remove from i-th intersection during q-th stage	$\delta_i^q(k) = \bar{\eta}_i \cdot \mathbf{p}_i^q \cdot \mathbf{x}_i(k)$
γ_i^q	number of vehicles which departed from the i-th intersection during the q-th stage (i.e. between the current time step k, and the end of the stage at time step $k + \tau_i^q(k)$)	$\gamma_i^q = \gamma_i^q(k, k + \tau_i^q(k))$

B.2.8 Tmin/Tmax Stage Duration Algorithm

Symbol	Description	Definition
$\bar{\tau}_{\min}$	minimum stage duration	-
$\bar{\tau}_{\max}$	maximum stage duration	-

For the Tmin/Tmax stage duration algorithm $\tau_i^q(k + \tau_i^q(k))$ is calculated s.t.,

$$\tau_i^q(k + \tau_i^q(k)) = \begin{cases} \frac{\tau_i^q(k) + \bar{\tau}_{\min}}{2} & \text{if } \delta_i^q(k) < \gamma_i^q(k + \tau_i^q(k)) \\ \frac{\tau_i^q(k) + \bar{\tau}_{\max}}{2} & \text{if } \delta_i^q(k) > \gamma_i^q(k + \tau_i^q(k)) \\ \tau_i^q(k) & \text{otherwise} \end{cases} \tag{B.12}$$

B.2.9 Proportional Stage Duration Algorithm

Symbol	Description	Definition
β_i^q	estimated error in stage duration	(see (B.13))
K_p	proportional gain	-

β_i^q is calculated s.t.,

$$\beta_i^q = \frac{\delta_i^q(k) - \gamma_i^q(k + \tau_i^q)}{\delta_i^q(k)} \cdot \tau_i^q(k) \tag{B.13}$$

For the Proportional stage duration algorithm $\tau_i^q(k + \tau_i^q(k))$ is calculated s.t.,

$$\tau_i^q(k + \tau_i^q(k)) = \tau_i^q \cdot \beta_i^q \cdot K_p \tag{B.14}$$

Bibliography

[1] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1575–1587, 2013.

[2] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos, "Store-and-forward based methods for the signal control problem in large-scale congested urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 2, pp. 163–174, 2009.

[3] K. Aboudolas, M. Papageorgiou, A. Kouvelas, and E. Kosmatopoulos, "A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 5, pp. 680–694, 2010.

[4] K. Aghabayk, M. Sarvi, and W. Young, "A State-of-the-Art Review of Car-Following Models with Particular Considerations of Heavy Vehicles," *Transport Reviews*, vol. 35, no. 1, pp. 82–105, 2015.

[5] R. Allsop, "SIGSET: A Computer Program for Calculating Traffic Signal Settings," *Traffic Engineering & Control*, vol. 13, no. 2, pp. 58–60, 1971.

[6] R. E. Allsop, "SIGCAP: A computer program for assessing the traffic capacity of signal-controlled road junctions," *Traffic Engineering and Control*, vol. 17, no. 819, pp. 338–341, 1976.

[7] M. K. Ardakani and L. Sun, "Decremental algorithm for adaptive routing incorporating traveler information," *Computers & Operations Research*, vol. 39, no. 12, pp. 3012–3020, dec 2012.

[8] J. Aslam, S. Lim, and D. Rus, "Congestion-aware Traffic Routing System using sensor data," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 1006–1013.

[9] T. Barker, G. Russo, and M. Di Bernardo, "A modular intersection controller with adaptive stage selection and duration algorithms," in *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2017 - Proceedings*, 2017, pp. 786–791.

[10] A. L. C. Bazzan, "A Distributed Approach for Coordination of Traffic Signal Agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 1, pp. 131–164, 2005.

[11] A. L. C. a. Bazzan, M. a. de Brito do Amarante, and F. B. B. Da Costa, "Management of Demand and Routing in Autonomous Personal Transportation," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 16, no. 1, pp. 1–11, 2012.

[12] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo (Simulation of Urban Mobility)," in *The Third International Conference on Advances in System Simulation*, 2011, pp. 55–60.

[13] M. G. Bell, "Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation," *Transportation Research Part B: Methodological*, vol. 43, no. 1, pp. 97–107, jan 2009.

[14] M. G. Bell, V. Trozzi, S. H. Hosseinloo, G. Gentile, and A. Fonzone, "Time-dependent Hyperstar algorithm for robust vehicle navigation," *Transportation Research Part A: Policy and Practice*, vol. 46, no. 5, pp. 790–800, jun 2012.

[15] M. E. Ben-Akiva, S. Gao, Z. Wei, and Y. Wen, "A dynamic traffic assignment model for highly congested urban networks," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 62–82, 2012.

[16] S. Box and B. Waterson, "An automated signalized junction controller that learns strategies by temporal difference reinforcement learning," *Engineering Applications of Artificial Intelligence*, 2012.

[17] D. Braess, A. Nagurney, and T. Wakolbinger, "On a Paradox of Traffic Planning," *Transportation Science*, vol. 39, no. 4, pp. 446–450, 2005.

[18] Cebr, "The future economic and environmental costs of gridlock in 2030," INRIX, Tech. Rep. July, 2014.

[19] I. Chabini and S. Lan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 60–74, mar 2002.

[20] C. S. Chang, *Performance Guarantees in Communication Networks.* London: Springer-Verlag, 2000.

[21] J. Cheng, W. Wu, J. Cao, and K. Li, "Fuzzy Group Based Intersection Control via Vehicular Networks for Smart Transportations," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–1, 2016.

[22] A. Chow, "System optimal traffic assignment with departure time choice," Ph.D. dissertation, University of London, 2007.

[23] R. Claes, T. Holvoet, and D. Weyns, "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 364–373, 2011.

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

[25] R. L. Cruz, "A calculus for network delay: Parts I and II," *IEEE Transactions on Information Theory*, vol. 37, no. I, pp. 114–141, 1991.

[26] C.S-W, "The Cost of Traffic Jams," 2014. [Online]. Available: http://www.economist.com/blogs/economist-explains/2014/11/economist-explains-1 [Accessed: 4th April 2017]

[27] C. F. Daganzo, "The Cell Transmission Model. Part I: A Simple Dynamic Representation of Highway Traffic," University of California, Berkeley, Tech. Rep., 1993.

[28] P. Dai, K. Liu, Q. Zhuge, E. H. Sha, V. Chung, S. Lee, and S. H. Son, "A Convex Optimization Based Autonomous Intersection Control Strategy in Vehicular Cyber-Physical Systems," in *Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, 2016, pp. 203–210.

[29] J. Dallmeyer, R. Schumann, A. D. Lattner, and I. J. Timm, "Don't go with the ant flow: Ant-inspired traffic routing in urban environments," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 19, no. 1, pp. 78–88, 2015.

[30] G. De Nunzio, G. Gomes, C. Canudas de Wit, R. Horowitz, and P. Moulin, "Arterial Bandwidth Maximization via Signal Offsets and Variable Speed Limits Control," *IEEE 54th Annual Conference on Decision and Control*, pp. 5142–5148, 2015.

[31] Department for Transportation, "Congestion: A National Issue," Tech. Rep. [Online]. Available: http://www.ops.fhwa.dot.gov/aboutus/opstory.htm

[32] C. Diakaki, V. Dinopoulou, A. Kostas, M. Papageorgiou, E. Ben-Shabat, E. Seider, and A. Leibov, "Extensions and New Applications of the Traffic-Responsive Urban Control Strategy - Coordinated Signal Control for Urban Networks," *Transportation Research Record*, vol. 1856, pp. 202–211, 2003.

[33] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2006.

[34] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, 2005.

[35] K. Dresner and P. Stone, "Multiagent Traffic Management : A Reservation-Based Intersection Control Mechanism," in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, no. July, New York, 2004, pp. 530–537.

[36] ——, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[37] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[38] M. Elhenawy, A. A. Elbery, A. A. Hassan, and H. A. Rakha, "An Intersection Game-Theory-Based Traffic Control Algorithm in a Connected Vehicle Environment," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2015-Octob, 2015, pp. 343–347.

[39] A. Ephremides and S. Verdu, "Control and Optimization Methods In Communication Network Problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 9, pp. 930–942, 1989.

[40] N. H. Gartner, "OPAC: A Demand Responsive Strategy for Traffic Signal Control," *Transportation Research Record*, vol. 906, no. January 1983, pp. 75–81, 1983.

[41] C. Gawron, "An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model," *International Journal of Modern Physics*, vol. 9, no. 3, pp. 393–407, 1998.

[42] D. C. . Gazis, "Traffic Flow Control : Theory and Applications," *American Scientist*, vol. 60, no. 4, pp. 414–424, 1972.

[43] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundation and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[44] P. G. Gipps, "A Behavioural Car Following Model for Computer Simulation," *Transportation Research Part B: Methodological*, vol. 15B, no. 2, pp. 105–111, 1981.

[45] R. Gordon and W. Tighe, "Traffic Control Systems Handbook," *Federal Highway Administration*, no. October, pp. 1–367, 2005.

[46] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2015.

[47] H. Guo, Z. Cao, M. Seshadri, J. Zhang, D. Niyato, and U. Fastenrath, "Routing Multiple Vehicles Cooperatively: Minimizing Road Network Breakdown Probability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 2, pp. 112–124, 2017.

[48] H. Guo, Z. Cao, J. Zhang, D. Niyato, and U. Fastenrath, "Routing Multiple Cars in Large Scale Networks : Minimizing Road Network Breakdown Probability," in *IEEE 17th International Conference on Intelligent Transportation Systems*, 2014, pp. 2180–2187.

[49] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative Collision Avoidance at Intersections: Algorithms and Experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, sep 2013.

[50] F. L. Hall, B. L. Allen, and M. A. Gunter, "Empirical Analysis of Freeway Flow-Density Relationships," *Transportation Research Part A: Policy and Practice*, vol. 20, no. 3, pp. 197–210, 1986.

[51] P. E. Hart and J. Nils, "Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[52] M. R. Hasan, A. L. C. Bazzan, E. Friedman, and A. Raja, "A Multiagent Solution to Overcome Selfish Routing In Transportation Networks," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1850–1855, 2016.

[53] J. Henry, J. Farges, and J. Tuffal, "the Prodyn Real Time Traffic Algorithm," *Control in Transportation Systems*, no. June, pp. 305–310, 1984.

[54] B. Heydecker, "Objectives, stimulus and feedback in signal control of road traffic," *Journal of Intelligent Transportation Systems*, vol. 8, no. 2, pp. 63–76, 2004.

[55] G. Improta and G. E. Cantarella, "Control system design for an individual signalized junction," *Transportation Research Part B*, vol. 18, no. 2, pp. 147–167, 1984.

[56] I. K. Isukapati and G. F. List, "Comparing actuated and bid-based control strategies," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1516–1521, 2016.

[57] O. Jahn, R. H. Mohring, A. S. Schulz, and N. E. Stier-Moses, "System-Optimal Routing of Traffic Flows with User Constraints in Networks with Congestion," *Operations Research*, vol. 53, no. 4, pp. 600–616, 2005.

[58] X. Ji, C. Shao, and B. Wang, "Stochastic Dynamic Traffic Assignment Model under Emergent Incidents," *Procedia Engineering*, vol. 137, pp. 620–629, 2016.

[59] B. Jiang, "A topological pattern of urban street networks: Universality and peculiarity," *Physica A: Statistical Mechanics and its Applications*, vol. 384, pp. 647–655, 2007.

[60] P. Kachroo and K. Ozbay, "Modeling of Network Level Traffic Routing Problem Using Nonlinear H Feedback Control," *Simulation*, vol. 10, no. 4, pp. 159–171, 2006.

[61] V. Kanagaraj, G. Asaithambi, C. N. Kumar, K. K. Srinivasan, and R. Sivanandan, "Evaluation of Different Vehicle Following Models Under Mixed Traffic Conditions," *Procedia - Social and Behavioral Sciences*, vol. 104, pp. 390–401, 2013.

[62] B. Kerner and H. Rehborn, "Experimental Properties of Phase Transitions in Traffic Flow," *Physical Review Letters*, vol. 79, no. 20, pp. 4030–4033, 1997.

[63] B. S. Kerner, "Physics of traffic gridlock in a city," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 84, no. 4, pp. 1–11, 2011.

[64] ——, "Breakdown minimization principle versus Wardrop's equilibria for dynamic traffic assignment and control in traffic and transportation networks: A critical mini-review," *Physica A: Statistical Mechanics and its Applications*, vol. 466, pp. 626–662, 2017.

[65] S. Krauss, "Microscopic Modelling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics," Ph.D. dissertation, University of Cologne, 1998.

[66] T. Le, P. Kovacs, N. Walton, H. L. Vu, L. L. H. Andrew, and S. S. P. Hoogendoorn, "Decentralized signal control for urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 431–450, 2015.

[67] W. Leutzbach and R. Wiedemann, "Development and Applications of Traffic Simulation Models at the Karlsruhe Intitut Fur Verkehrwesen," *Traffic Engineering & Control*, vol. 27, no. 5, pp. 270–278, 1986.

[68] M. W. Levin, H. Fritz, and S. D. Boyles, "On Optimizing Reservation-Based Intersection Controls," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 505–515, 2016.

[69] H. Lieu, "Traffic-Flow Theory." *Public Roads*, vol. 62, no. 4, pp. 45–47, 1999.

[70] M. J. Lighthill and G. B. Whitham, "On kinematic waves II . A theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[71] S. Lim and D. Rus, "Stochastic distributed multi-agent planning and applications to traffic," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2012, pp. 2873–2879.

[72] R. Ma, X. J. Ban, and J. S. Pang, "Continuous-time dynamic system optimum for single-destination traffic networks with queue spillbacks," *Transportation Research Part B: Methodological*, vol. 68, pp. 98–122, 2014.

[73] M. Mahut, "An heuristic algorithm for simulation-based dynamic traffic assignment," *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pp. 239–244, 2005.

[74] A. P. Masucci, D. Smith, A. Crooks, and M. Batty, "Random planar graphs and the London street network," *European Physical Journal B*, vol. 71, no. 2, pp. 259–271, 2009.

[75] D. K. Merchant and G. L. Nemhauser, "A Model and an Algorithm for the Dynamic Traffic Assignment Problem," *Transportation Science*, vol. 12, no. 3, pp. 183–199, 1978.

[76] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[77] R. Mohan and G. Ramadurai, "State-of-the art of macroscopic traffic flow modelling," *International Journal of Advances in Engineering Sciences and Applied Mathematics*, vol. 5, no. 2-3, pp. 158–176, 2013.

[78] K. Nagel and M. Schreckenberg, "A Cellular Automaton Model for Freeway Traffic," *Journal de Physique I France*, vol. 2, pp. 2221–2229, 1992.

[79] M. E. J. Newman, "The structure and function of complex networks," *Society for Industrial and Applied Mathematics*, vol. 45, no. 2, pp. 167–256, 2003.

[80] H. Papadimitriou, "the Euclidean Traveling Salesman Problem is NP-Complete," *Theoretical Computer Science*, vol. 4, pp. 237–244, 1977.

[81] M. Papageorgiou, "Dynamic Modelling, Assignment, and Route Guidance in Traffic Networks," *Transportation Research Part B: Methodological*, vol. 240, no. 6, pp. 471–495, 1990.

[82] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of Road Traffic Control Strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2008.

[83] F. Perronnet, A. Abbas-turki, and A. E. Moudni, "Vehicle Routing through Deadlock-free Policy for Cooperative Traffic Control in a Network of Intersections: Reservation and Congestion," in *IEEE 17th International Conference on Intelligent Transportation Systems*, 2014, pp. 2233–2238.

[84] A. Philbrick, "A Short History of the Development of the Gravity Model," *Australian Road Research*, vol. 5, no. 4, pp. 40–54, 1973.

[85] J. S. J. Ren, W. Wang, and S. S. Liao, "Optimal Control Theory in Intelligent Transportation Systems Research - A Review," City University of Hong Kong, Hong Kong, Tech. Rep., 2013. [Online]. Available: http://dblp.uni-trier.de/rec/bib/journals/corr/abs-1304-3778

[86] P. I. Richards, "Shock Waves on the Highway," *Operations Research*, vol. 4, no. 1, pp. 42–51, 1956.

[87] S. Richmond, "Google threatens satnav with real-time traffic data," 2012. [Online]. Available: http://www.telegraph.co.uk/technology/google/9175758/Google-threatens-satnav-with-real-time-traffic-data.html [Accessed: 20th September 2016]

[88] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signals in real time–The SCOOT method," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1 pt 1, pp. 11–15, 1991.

[89] D. Robertson, "TRANSYT: A Traffic Network Study Tool," *Road Research Laboratory*, vol. 253, 1969.

[90] T. Roughgarden and E. Tardos, "How bad is selfish routing?" *Proceedings 41st Annual Symposium on Foundations of Computer Science*, vol. 49, no. 2, pp. 1–26, 2000.

[91] A. Sadek, B. Smith, and M. Demetsky, "Dynamic Traffic Assignment: Genetic Algorithms Approach," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1588, no. 971192, pp. 95–103, 1997.

[92] M. Schwager, J. Mclurkin, and D. Rus, "Distributed Coverage Control with Sensory Feedback for Networked Robots," *Proceedings of Robotics: Science and Systems*, pp. 49–56, 2006.

[93] Y. Sheffi, *Urban Transportation Networks*. Englewoord Cliffs: Prentice-Hall, 1984.

[94] A. G. Sims and K. W. Dobinson, "The Sydney Coordinated Adaptive Traffic (SCAT) System Philosophy and Benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[95] Z. Sitavancová and M. Hájek, "Intelligent Transport Systems Thematic Summary European Commission," European Commission DG Energy and Transport, Tech. Rep., 2009. [Online]. Available: http://www.transport-research.info/Upload/Documents/201002/20100215_125401_19359_TRS_IntelligentTransportSystems.pdf [Accessed: 2015-1-10]

[96] H. Spiess and M. Florian, "Optimal strategies: A new assignment model for transit networks," *Transportation Research Part B: Methodological*, vol. 23, no. 2, pp. 83–102, 1989.

[97] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, "Revisiting Street Intersections Using Slot-Based Systems," *PLOS ONE*, vol. 11, no. 3, pp. 1–9, 2016.

[98] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Polocies for Max Throughput in Multihop Wireless Networks.Pdf," *IEEE Conference on Decision and Control*, vol. 31, no. 1, pp. 2130–2132, 1990.

[99] M. A. P. Taylor and W. Young, *Traffic Analysis: New Technology & New Solutions*. Melbourne: Hargreen Publishing Company, 1988.

[100] N. B. Taylor and B. G. Heydecker, "The effect of green time on stochastic queues at traffic signals," *Transportation Planning and Technology*, vol. 37, no. 1, pp. 3–19, oct 2014.

[101] The Economist, "The Future of Personal Transport: The driverless, car-sharing road ahead," 2016. [Online]. Available: https://www.economist.com/news/business/21685459-carmakers-increasingly-fret-their-industry-brink-huge-disruption [Accessed: 2nd April 2017]

[102] TomTom International, "TomTom European Traffic Index," TomTom International, Tech. Rep., 2014. [Online]. Available: http://www.tomtom.com/en_gb/trafficindex/#/ [Accessed: 2016-10-10]

[103] ——, "TomTom Traffic Index Web Page," 2017. [Online]. Available: https://www.tomtom.com/en_gb/trafficindex/list?citySize=LARGE&continent=ALL&country=ALL [Accessed: 2016-10-10]

[104] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics (SIAM), 2002.

[105] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2014.

[106] P. Varaiya, "The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections," in *Advances in Dynamic Network Modeling in Complex Transportation Systems*. Springer, 2013, vol. 2, ch. 2, pp. 225–244.

[107] W. S. Vickrey, "Congestion theory and transportation investment," *The American Economic Review*, vol. 59, no. 2, pp. 251–260, 1969.

[108] D. Wagner and T. Willhalm, "Speed-Up Techniques for Shortest-Path Computations," *STACS 2007*, vol. 4393, pp. 23–36, 2007.

[109] N. Wagner, "The dynamic user equilibrium on a transport network : mathematical properties and economic applications," Ph.D. dissertation, University of Paris-Est, 2014.

[110] S. Wang, S. Djahel, and J. Mcmanis, "An Adaptive and VANETs-based Next Road Re-routing System for Unexpected Urban Traffic Congestion Avoidance," in *IEEEE Vehicular Networking Conference (VNC)*, 2015, pp. 196–203.

[111] ——, "A Multi-Agent Based Vehicles Re-routing System for Unexpected Traffic Congestion Avoidance," in *IEEE 17th International Conference on Intelligent Transportation Systems*, 2014, pp. 2541–2548.

[112] S. Wang, S. Djahel, J. McManis, C. McKenna, and L. Murphy, "Comprehensive performance analysis and comparison of vehicles routing algorithms in smart cities," in *Global Information Infrastructure Symposium*, 2013, pp. 1–8.

[113] J. G. Wardrop, "Some Theoretical Aspects of Road Traffic Research," *ICE Proceedings: Engineering Divisions*, vol. 1, no. 5, pp. 767–768, 1952.

[114] F. V. Webster, *Traffic signal settings*. London: H.M.S.O., 1958.

[115] D. Wilkie, J. van den Berg, M. Lin, and D. Manocha, "Self-Aware Traffic Route Planning," *Association for the Advancement of Artificial Intelligence*, pp. 1521–1527, 2011.

[116] T. Wongpiromsarn, T. Uthaicharoenpong, Y. Wang, E. Frazzoli, and D. Wang, "Distributed traffic signal control for maximum network throughput," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 588–595, 2012.

[117] ——, "Distributed traffic signal control for maximum network throughput," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 588–595, 2012.

[118] N. Xiao, Y. Li, Y. Luo, E. Frazzoli, Y. Wang, and D. Wang, "Vissim simulation for extended back-pressure traffic signal control strategy," in *Proceedings of the 14th Intelligent Transport System Asia Pacific Forum (ITS-AP)*, 2015.

[119] N. Xiao, E. Frazzoli, Y. Li, Y. Luo, Y. Wang, and D. Wang, "Throughput Optimality of Extended Back-pressure Traffic Signal Control Algorithm," in *Mediterranean Conference on Control and Automation*, 2015, pp. 1059–1064.

[120] J. Y. Yen, "An Algorithm for Finding Shortest Routes from All Source Nodes to a given Destination in General Networks," *Quarterly of Applied Mathematics*, vol. 27, pp. 526–530, 1970.

[121] W. Yin and X. Yang, "A Totally Astar-based Multi-path Algorithm for the Recognition of Reasonable Route Sets in Vehicle Navigation Systems," in *13th COTA International Conference of Transportation Professionals*, vol. 96. Elsevier B.V., nov 2013, pp. 1069–1078.

[122] G. Zhang, "Measuring the efficiency of network designing," in *First International Conference, Complex*, 2009, pp. 503–513.

[123] A. K. Ziliaskopoulos, "Foundations of Dynamic Traffic Assignment : The Past , the Present and the Future," *Networks and Spatial Economics*, vol. 1, no. 3, pp. 233–265, 2001.