UNIVERSITY OF CALIFORNIA, SAN DIEGO

Design and Analysis of Distributed Algorithms with Applications to Networked Traffic Systems

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

in

Engineering Sciences (Mechanical Engineering)

by

Evan Gravelle

Committee in charge:

Professor Sonia Martínez, Chair
Professor Jorge Cortés
Professor Mauricio de Oliveira
Professor Massimo Franceschetti
Professor Melvin Leok

2017

The dissertation of Evan Gravelle is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

Signature Page iii

Table of Contents iv

List of Figures vi

Acknowledgements vii

Vita ix

Abstract of the Dissertation x

Chapter 1 Introduction 1
1.1 Outline 2
1.2 Notation 3
1.3 Graph Theory 3

Chapter 2 An Anytime Distributed Load Balancing Algorithm Satisfying Capacity and Quantization Constraints 4
2.1 Abstract 4
2.2 Introduction 5
2.2.1 Contributions 6
2.2.2 Outline 7
2.3 Problem statement 7
2.3.1 Characterization of balanced states 8
2.3.2 Main theorem 12
2.4 Load Balancing Algorithm 13
2.5 Convergence Analysis 14
2.6 Simulations 32
2.6.1 Example Problem 32
2.6.2 Convergence Rate 32
2.7 Conclusions and Future Work 34
2.8 Additional Proofs 35
2.8.1 Proof that lines of passes without capped nodes always cause a decrease in W 35
2.8.2 Proof that lines of passes including maximum capacities always cause a decrease in $\hat{W}$ 39
2.8.3 Proof of destruction of coin in finite time 42

Chapter 3 Distributed Dynamic Lane Reversal and Rerouting for Traffic Delay Reduction . . . 50
3.1 Abstract . . . 50
3.2 Introduction . . . 51
3.2.1 Contributions . . . 53
3.2.2 Outline . . . 53
3.3 Problem Statement . . . 53
3.3.1 Traffic Model . . . 54
3.3.2 Problem Formulation . . . 58
3.3.3 Approximation of the State . . . 60
3.4 Lane Reversal Policy . . . 61
3.4.1 Lane Reversal Algorithm . . . 61
3.4.2 Stability Analysis of Lane Reversal . . . 63
3.5 Vehicle Rerouting Policy . . . 64
3.5.1 Stability Analysis of Weight-Balanced Traffic Networks 65
3.5.2 Rerouting Algorithm . . . 68
3.6 Simulation Results . . . 70
3.6.1 Lane Reversal Algorithm . . . 70
3.6.2 Rerouting Algorithm . . . 72
3.7 Conclusion and Future Work . . . 74

Chapter 4 Using time-inconsistent wait-time functions for cycle-free coordinated traffic intersections . . . 76
4.1 Abstract . . . 76
4.2 Introduction . . . 77
4.2.1 Contributions . . . 78
4.2.2 Outline . . . 79
4.3 Problem Formulation . . . 80
4.3.1 Single intersection problem . . . 80
4.3.2 Multiple intersection problem . . . 82
4.3.3 Vehicle Behavior Model . . . 85
4.3.4 Wait-Time Function . . . 85
4.4 Queue Stability . . . 86
4.5 Simulation Results . . . 91
4.6 Conclusion and Future Work . . . 95

Chapter 5 Closing Remarks . . . 97

Bibliography . . . 99

LIST OF FIGURES

Figure 2.1: Example connected graph 33
Figure 2.2: Initial load distribution 34
Figure 2.3: Final load distribution 35
Figure 2.4: Plot of $W(x)$ over k_f iterations 36
Figure 2.5: Plot of average convergence time 37

Figure 3.1: Road divided into cells 54
Figure 3.2: Vehicle flow model 55
Figure 3.3: Vehicles before lane reversal 58
Figure 3.4: Vehicles after lane reversal 58
Figure 3.5: Schedule of road network 62
Figure 3.6: Flow rate using lane reversal on two roads 72
Figure 3.7: Lane reversal and rerouting performance 72
Figure 3.8: Effect of noise on equilibrium 73
Figure 3.9: Road network density 73
Figure 3.10: Flow rate using lane reversal on two roads 74

Figure 4.1: Plot of coordination term 84
Figure 4.2: Image of microscopic model. 92
Figure 4.3: Plot of max queue length 93
Figure 4.4: Evolution of weight per phase under custom algorithm 93
Figure 4.5: Evolution of weight per phase under cyclical algorithm 94
Figure 4.6: Performance of algorithms, in kiloseconds 94
Figure 4.7: Plot of performance over coordination coefficient 95
Figure 4.8: Zoomed plot of performance over coordination coefficient 96

ACKNOWLEDGEMENTS

First of all, I'd like to thank my advisor Sonia Martínez for sharing her wealth of knowledge of controls and mathematics with me, for being patient when I am slow to progress, and for staying up and working weekends to finish papers on time.

To my thesis committee Jorge Cortés, Massimo Franceschetti, Melvin Leok, Sonia Martínez, and Maurício de Oliveira, thank you for your valuable feedback on this dissertation and my defense.

To Mike Ouimet, thanks for teaching me all kinds of nerdy things, and for inspiring me to learn about artificial intelligence. To Aaron Ma, thanks for going above and beyond in every robotics project we have done together.

To Tor Anderson, Beth Boardman, Chin-Yao Chang, Andrés Cortés, Ashish Cherukuri, Solmaz Kia, Vishaal Krishnan, Dan Li, David Mateos, Erfan Nozari, Pio Ong, Eduardo Ramirez, Dean Richert, Priyank Srivastava, Aamodh Suresh, Pavan Tallapragada, and Yifu Zhang, thanks for the engaging conversations and making our group outstanding.

To my parents, thanks for always supporting me, for raising me to love learning new things, and for all of your advice. To Caitlin, thanks for the helpful paper edits and for putting up with me when I'm stressed. This work was funded by NSF-CMMI 1434819 and AFOSR 11RSL548.

Chapter 2, in part, has been published in proceedings of IEEE Int. Conf. on Decision and Control, Los Angeles, CA, USA, December 2014, by E. Gravelle and S. Martínez, and in IEEE Transactions on Control of Networked Systems, 4(2):279–287, 2017, by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, has been published in proceedings of Mathematical Theory of Networks and Systems, Minneapolis, MN, USA, July 2016, by E. Gravelle and S. Martínez, and is submitted for publication to International Journal of Control, 2017,

by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is submitted for publication to IEEE Conf. on Control Technology and Applications, 2017, by E. Gravelle and S. Martínez and IET Control Theory & Applications, 2017, by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

VITA

2012	Bachelor of Science in Mechanical Engineering, University of California, Santa Barbara
2012-2013	Graduate Student Researcher, University of California, San Diego
2013	Master of Science in Engineering Sciences (Mechanical Engineering), University of California, San Diego
2013-2016	Graduate Student Researcher, University of California, San Diego
2017	Doctor of Philosophy in Engineering Sciences (Mechanical Engineering), University of California, San Diego

PUBLICATIONS

Journal papers:

E. Gravelle and S. Martínez. *Using time-inconsistent wait-time functions for cycle-free coordinated traffic intersections*. Submitted June 2017 to IET Control Theory & Applications.

E. Gravelle and S. Martínez. *Distributed Dynamic Lane Reversal and Rerouting for Traffic Delay Reduction*. Accepted May 2017 to International Journal of Control.

E. Gravelle and S. Martínez. *An Anytime Distributed Load Balancing Algorithm Satisfying Capacity and Quantization Constraints*. IEEE Transactions on Control of Networked Systems, 2017.

Conference papers:

E. Gravelle and S. Martínez. *A cycle-free coordinated traffic intersection policy using time-inconsistent wait-time functions*. Accepted to 2017 IEEE Conference on Control Technology and Applications.

E. Gravelle and S. Martínez. *Traffic Delay Reduction via Distributed Dynamic Lane Reversal and Rerouting*. 22nd International Symposium on Mathematical Theory of Networks and Systems, July 2016.

E. Gravelle and S. Martínez. *Quantized Distributed Load Balancing with Capacity Constraints*. IEEE Conference on Decision and Control, December 2014.

ABSTRACT OF THE DISSERTATION

Design and Analysis of Distributed Algorithms with Applications to Networked Traffic Systems

by

Evan Gravelle

Doctor of Philosophy in Engineering Sciences (Mechanical Engineering)

University of California, San Diego, 2017

Professor Sonia Martínez, Chair

There are many benefits of solving problems in a decentralized manner. Distributed algorithms often do not require global information which can alleviate the curse of dimensionality in large networks, there is often robustness to failure of parts, and they are often more robust to failure of parts, and to dynamic changes to the environment that can occur while maintaining performance. This dissertation will focus on three problems involving networked systems in which distributed algorithms have significant benefits: constrained load balancing, traffic congestion minimization, and traffic intersection efficiency.

Many physical limitations of real systems are not considered in the literature of distributed load balancing algorithms. We address the specific problem of quantized distributed load balancing over a network of agents subject to upper-limit constraints. We then shift focus to traffic systems, where endowing traffic control systems with local information and communication can be exploited for further efficiency. Motivated by a desire to reduce congestion, we propose two distributed algorithms to reduce delays: a dynamic lane reversal algorithm and a rerouting algorithm. Finally, we present a novel intersection control algorithm based on an objective function that accounts for drivers' time preferences.

For each problem, a specific objective is formed mathematically. An algorithm is designed to achieve this objective, and stability and convergence of the algorithms are analyzed. Experiments are run through simulation to verify stability and convergence as well as to test performance.

Chapter 1

Introduction

Modern technology has made the existence of cyber-physical systems common in society, and these systems will be increasingly more common in the near future. Cyber-physical systems integrate computer algorithms and communication with a physical system and users. One example is seen in traffic systems. The advancement of autonomous cars has removed the requirement of a human driver, and this has far-reaching consequences. Transportation costs will greatly decrease due to the existence of driverless taxis. Traffic congestion will be greatly reduced due to more conservative driving of autonomous vehicles, there will far less accidents, there can be less cars on the road due to easier vehicle sharing, and traffic intersections can be redesigned in a much more efficient way to take advantage of vehicle autonomy, greater quality and quantity of information, and more intelligent algorithms. To achieve this, vehicles and intersection controllers require high amounts of information, algorithms to process this information and make intelligent decisions, and methods to communicate with other agents in order for the system to benefit as a whole. Beyond traffic systems, similar gains can be found in air and sea travel, manufacturing, agriculture, and power networks.

However, there is much research to be done to make this a reality. The complexity

of these systems makes many centralized algorithms intractable, so responsibilities must be distributed, and only local information should be used whenever possible. Safety guarantees should be made when the systems involve humans, required thorough analysis of the system dynamics. Performance should be weighed against cost and computation to accomplish each objective. This dissertation provides the work we have done towards this end.

1.1 Outline

- *Chapter 2*: This chapter focuses on the load balancing problem over a network. We formulate the problem, provide a distributed load balancing algorithm to balance the loads, analyze the convergence properties of the algorithm based on graph conditions, and examine performance in simulation.

- *Chapter 3*: This chapter focuses on the traffic lane reversal and rerouting problems. We formulate the macroscopic traffic model and lane reversal/rerouting objectives, provide a lane reversal algorithm and rerouting algorithm, and examine performance in simulation.

- *Chapter 4*: This chapter focuses on the traffic intersection efficiency problem. We formulate the microscopic model and the intersection objective, provide an algorithm to optimize the objective, analyze the convergence properties of the algorithm, and examine performance in simulation.

- *Closing Remarks*: This chapter will conclude the dissertation.

1.2 Notation

This section presents some basic mathematical notations and concepts used throughout the thesis.

We let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_{\geq 0}$ denote the set of non-negative real numbers, $\mathbb{N}$ denote the set of natural numbers, and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Similarly, $\mathbb{R}^n$ (resp. $\mathbb{R}^n_{\geq 0}, \mathbb{N}^n, \mathbb{N}^n_0$) denotes the product space of n copies of $\mathbb{R}$ (resp. $\mathbb{R}_{\geq 0}$, $\mathbb{N}, \mathbb{N}_0$). The $n \times n$ identity matrix is designated by I_n. The transpose of matrix B is denoted by $B^\top$. The n-dimensional vector of ones is represented as $\mathbf{1}_n = [1, \ldots, 1]^\top$. The cardinality of set A is denoted by $|A|$. The floor (resp. ceiling) operator on $\mathbb{R}$ is denoted by $\lfloor \cdot \rfloor$ (resp. $\lceil \cdot \rceil$) : $\mathbb{R}_{\geq 0} \to \mathbb{N}_0$. Assigning the value y to a variable x is denoted $x \leftarrow y$. We define $\mathbf{i} = \sqrt{-1}$. Matrix $A = \{a_{ij}\}$ satisfies $A \in \text{sparse}(G)$, for $G = (V, E)$, if $a_{ij} = 0$ when $(i, j) \notin E$.

1.3 Graph Theory

We now summarize some basic notions from algebraic graph theory. A *directed graph* G consists of a set of *vertices* V and a set of *directed edges* E, $G = (V, E)$, such that $E \subset V \times V$. Vertex a is an *out-neighbor* of vertex b if $(b, a) \in E$. Similarly, a is an *in-neighbor* of b if $(a, b) \in E$. Vertex a is a neighbor of b if b is an out-neighbor or in-neighbor of a. The set of out-neighbors (resp. in-neighbors) of a is denoted $\mathcal{N}_a^{\text{out}}$ (resp. $\mathcal{N}_a^{\text{in}}$). A graph is *undirected* if $\forall (i, j) \in E$, $(j, i) \in E$. For undirected graphs, $\mathcal{N}_i$ denotes the set of neighbors of node i in V. The graph G is *connected* if for any pair of nodes (i, j) there exists a sequence of edges $(i, a_1), (a_1, a_2), \ldots, (a_m, j)$ connecting i with j. A node i is within *d-hops* of j if there exists a sequence of d or less edges connecting i with j [5].

Chapter 2

An Anytime Distributed Load Balancing Algorithm Satisfying Capacity and Quantization Constraints

2.1 Abstract

Current research in the field of distributed consensus algorithms fails to adequately address physical limitations of real systems. In this chapter we propose a new algorithm for quantized distributed load balancing over a network of agents subject to upper-limit constraints. More precisely, each node has a load that takes an integer value and is constrained to remain under its maximum load capacity at all times. Convergence to a set of balanced states is proven for all connected graphs with any feasible initial load distribution, given some conditions on the placement of nodes with finite capacity. We present simulations that verify our results and discuss possible extensions of the algorithm.

2.2 Introduction

An important problem in distribution and service networks consists of splitting tasks among a group of providers or nodes which can offer certain network commodities to users. Examples include task distribution among a set of processors or robots, energy provision by a set of generators, or traffic control by a group of network managers. The nodes in the network may be subject to capacity constraints, which can restrict the amount of load they can handle, e.g. a finite memory bank in a processor, a maximum generating capacity of a generator, or a maximum road capacity in a traffic network. Another potential restriction is the indivisibility of jobs that need be completed by nodes in the network. Motivated by this, we design and analyze a new distributed algorithm that can achieve load balancing while satisfying capacity and indivisible load constraints. If this algorithm is interrupted prior to completion, the state will satisfy all constraints and will generally be more balanced.

The problem of distributed consensus or agreement has received much attention in recent years. The problem has roots in parallel computation [4] and has been addressed in complementary ways in [25], [36], and [6]. Load balancing is a constrained version of the consensus problem, and gossip-based algorithms for distributed load balancing over different types of graphs with various constraints and objectives have been proposed [18], [19]. More recently, there has been a focus on quantized consensus to address the problems of finite capacity in communication channels and finite precision in computation. An example of this work studies distributed gossip algorithms which converge to an integer approximation of the average of the initial values and satisfy the integer constraint at all times [26]. An efficient method of quantization using symbols instead of numbers as exchanged data in solving the distributed average consensus problem is seen in [9]. Dithered forms of quantization are also seen in [3]. An algorithm solving the distributed

quantized consensus problem over heterogeneous networks which also minimizes time given different processor speeds is found in [20]. Quantization under a distributed averaging algorithm that does not conserve the initial sum can be found in [33].

The convergence rate of gossip-based distributed quantized consensus algorithms is studied in [21] while convergence rate for non-gossip based algorithms is studied in [51], [29]. Fanti et al. [17] consider discrete consensus in networks with constrained capacity, where nodes have the same finite capacity, tasks have different costs, and initial states are infeasible. Employing a gossip communication scheme, this work focuses on finding an initial feasible task assignment, which minimizes the total load assigned to agents. Gossip-based algorithms are often easy to analyze but in general do not converge as quickly as synchronous algorithms.

2.2.1 Contributions

Here, we design a new distributed algorithm to balance quantized loads among nodes which can be subject to maximum load capacities. A node filled to capacity behaves differently from a node that is below its capacity, and this distinction is crucial for convergence of the algorithm. Nodes which reach their maximum capacity introduce an effect similar to state-dependent delay, so a major difference between our algorithm and standard quantized consensus algorithms is that these nodes can make passes which unbalance the network, significantly complicating the analysis. We prove convergence of this algorithm to a set of balanced states assuming certain conditions on the graph and on the distribution of nodes subject to maximum capacities. We then simulate the algorithm over a variety of graphs and initial load configurations to test our results, one example is provided.

2.2.2 Outline

The chapter is organized as follows. In Section 2.3, we characterize the specific problem we wish to solve and the solution set that we wish to converge to. In Section 2.4, we provide our load balancing algorithm. In Section 2.5, we prove convergence of our algorithm to our desired solution set under some basic assumptions. In Section 2.6, we simulate our algorithm and provide some results on convergence rate. In Section 2.7, we summarize our results and discuss possible ways to extend our work. In Section 2.8, we provide some additional proofs.

2.3 Problem statement

In this section we define the system and describe the objective. Consider a network of n agents with communication and interaction described by an undirected graph $G = (V, E)$. The state of the network, denoted by $x = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{N}_0^n$, is given by the total load at each node, which is initially $x(0)$. Each node i has a maximum load capacity $c_i \in \mathbb{N} \cup \{\infty\}$, constraining $x_i(k) \leq c_i$ for all times $k \in \mathbb{N}$.

The objective of the network is to distribute loads as evenly as possible across the nodes of the network. This translates into achieving a state that is a solution to the following minimization problem, Problem (2.1):

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & W(x) = \sum_{i \in V} (x_i - \overline{x})^2, \\ \text{subject to} \quad & \sum_{i \in V} x_i = n\overline{x}, \\ & x_i \leq c_i, \ \forall\, i \in V, \\ & x_i \in \mathbb{N}_0, \ \forall\, i \in V, \end{aligned} \tag{2.1}$$

where $\overline{x} = \frac{1}{n}\mathbf{1}_n^\top x_i(0)$. This function $W(x)$ is the *variance* of x, which measures the distance between x and $\overline{x}\mathbf{1}_n$. A state is *feasible* if it satisfies all constraints.

2.3.1 Characterization of balanced states

We will characterize the set of solutions to Problem (2.1). If a state x is in this solution set then we call it a *balanced state*. Assuming some initial state $x(0)$, the following lemma will be useful:

Lemma 3.1. *If there exists* $p \in \operatorname{argmin}_{i\in V} c_i$ *with* $c_p \le \lfloor \overline{x} \rfloor$*, then any solution to Problem* (2.1) *requires* $x_p = c_p$.

Proof. Suppose $x_p < c_p$ with $c_p \le \lfloor \overline{x} \rfloor$ in some feasible state x for some node $p \in V$, and define a new state x^+ by updating two nodes, $x_p^+ = x_p + 1$ and $x_q^+ = x_q - 1$ for some $q \in \operatorname{argmax}_{i\in V} x_i$, and holding the rest constant.

For some node $q \in \operatorname{argmax}_{i\in V} x_i$, $x_q^+ = x_q - 1$ is a feasible state for q because $c_p \le \lfloor \overline{x} \rfloor$ and $x_p < c_p$ implies there is some q such that $x_q \ge 1$. Since $x_p < c_p$, then $x_p^+ \le c_p$, so x_p is feasible. Define $\Delta W = W(x^+) - W(x)$. Then,

$$\begin{aligned} \Delta W &= (x_q^+ - \overline{x})^2 + (x_p^+ - \overline{x})^2 - (x_q - \overline{x})^2 - (x_p - \overline{x})^2 \\ &= 2(x_p - x_q + 1). \end{aligned} \tag{2.2}$$

So if $x_q > x_p + 1$, $\Delta W < 0$. To see that this holds,

- Suppose $\lfloor \overline{x} \rfloor = \overline{x}$. Then, $n\overline{x} = n\lfloor \overline{x} \rfloor = \mathbf{1}_n^\top x(0)$.

  Since $x_p < \overline{x}$, then there exists q' s.t. $x_{q'} > \overline{x}$ with $x_{q'} \le x_q$, so $x_q > x_p + 1$.

- Suppose $\lfloor \overline{x} \rfloor < x$. There must exist q' s.t. $x_{q'} \ge \lceil \overline{x} \rceil = \lfloor \overline{x} \rfloor + 1$. Since $x_p + 1 \le \lfloor \overline{x} \rfloor \le x_{q'} - 1 \le x_q - 1$, then $x_q > x_p + 1$.

In summary, for some $p \in \operatorname{argmin}_{i \in V_c} c_i$, if $c_p \leq \lfloor \overline{x} \rfloor$ and $x_p < c_p$ then we define $x_p^+ = x_p + 1$ and $x_q^+ = x_q - 1$ which results in $W(x^+) < W(x)$. After $c_p - x_p$ iterations of this, we have $x_p^+ = c_p$, so we can say that a state that minimizes $W(x)$ when $c_p \leq \lfloor \overline{x} \rfloor$ must include $x_p = c_p$. □

Apply Lemma 3.1 to $x(0)$, and if such a p exists, pick one and define $\tilde{V}_1 = \{p\}$, $V_1 = V \setminus \tilde{V}_1$, and $\overline{x}_1 = \dfrac{n\overline{x} - c_p}{n-1}$, where $\overline{x}_1$ is the new average load excluding p. With a slight abuse of notation, we redefine x as the vector of remaining free states (here and later on). Our problem now reduces to

$$\begin{aligned} \underset{x \in \mathbb{R}^{n-1}}{\text{minimize}} \quad & W(x) = \sum_{i \in V_1} (x_i - \overline{x})^2, \\ \text{subject to} \quad & \sum_{i \in V_1} x_i = (n-1)\overline{x}_1, \\ & x_i \leq c_i, \ \forall\, i \in V_1, \\ & x_i \in \mathbb{N}_0, \ \forall\, i \in V_1. \end{aligned} \tag{2.3}$$

Lemma 3.1 can be applied to this new problem with reduced state. Define a new $p \in \operatorname{argmin}_{i \in V_1} c_i$. If $c_p \leq \lfloor \overline{x}_1 \rfloor$ then we know the solution contains $x_p = c_p$. We repeat this process for ℓ iterations, defining $\tilde{V}_\ell = \{p\} \cup \tilde{V}_{\ell-1}$, $V_\ell = V \setminus \tilde{V}_\ell$, and $\overline{x}_\ell = \dfrac{(n-\ell+1)\overline{x}_{\ell-1} - c_p}{n-\ell}$, until we have $\min_{i \in V_\ell} c_i > \lfloor \overline{x}_\ell \rfloor$. This results in $\tilde{V}_\ell = \{i \in V \mid c_i \leq \lfloor \overline{x}_\ell \rfloor\}$. After defining $m = |V_\ell|$ and labeling our remaining node indices from 1 to m, Problem (2.3) reduces to

$$\begin{aligned} \underset{x \in \mathbb{R}^{m}}{\text{minimize}} \quad & W(x) = \sum_{i=1}^{m} (x_i - \overline{x})^2, \\ \text{subject to} \quad & \sum_{i=1}^{m} x_i = m\overline{x}_\ell, \\ & x_i \leq c_i, \ \forall\, i \in V_\ell, \\ & x_i \in \mathbb{N}_0, \ \forall\, i \in V_\ell, \end{aligned} \tag{2.4}$$

and we know that $c_i > \lfloor \overline{x}_\ell \rfloor$, $\forall\, i \in V_\ell$. To solve Problem (2.4) we will first temporarily relax the last two constraints so we are left with

$$\begin{aligned} \underset{x\in\mathbb{R}^m}{\text{minimize}} \quad & W(x) = \sum_{i=1}^{m} (x_i - \overline{x})^2, \\ \text{subject to} \quad & \sum_{i=1}^{m} x_i = m\overline{x}_\ell. \end{aligned}$$

We can reduce dimensionality and eliminate the last constraint by substituting $x_m = m\overline{x}_\ell - \sum_{i=1}^{m-1} x_i$, so we are left with

$$\underset{x\in\mathbb{R}^{m-1}}{\text{minimize}} \quad W(x) = \left(m\overline{x}_\ell - \sum_{i=1}^{m-1} x_i - \overline{x} \right)^2 + \sum_{i=1}^{m-1} (x_i - \overline{x})^2.$$

The partial derivitive is

$$\frac{\partial W}{\partial x_j}(x) = 2x_j + 2\sum_{i=1}^{m-1} x_i - 2m\overline{x}_\ell,\ j \in \{1, \ldots, m-1\},$$

and the Hessian is

$$H = \frac{\partial W}{\partial x_i \partial x_j}(x) = \begin{bmatrix} 4 & 2 & \cdots & 2 \\ 2 & 4 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 2 & 2 & \cdots & 4 \end{bmatrix}.$$

Note that H is a circulant matrix with eigenvalues given by

$$\begin{aligned} \lambda_j &= 4 + 2\omega_j + 2\omega_j^2 + \ldots + 2\omega_j^{n-1}, \\ \omega_j &= e^{\frac{2\pi \mathbf{i} j}{n}}, \quad \forall\, j \in \{0, 1, \ldots, n-1\}, \end{aligned}$$

where $\{\omega_0, \omega_1, \ldots, \omega_{n-1}\}$ are the n-th roots of unity, which satisfy ${\omega_j}^n - 1 = 0$. In particular, $\omega_0 = 1$ implies $\lambda_0 = 2n + 2$.

If $n \geq 2$, then for $j \in \{1, \ldots, n-1\}$, $\omega_j = e^{\frac{2\pi \mathbf{i} j}{n}} \neq 1$. We can rewrite $\omega^n - 1 = (\omega -$

$1)(1+\omega+\ldots+\omega^{n-1})=0$. Since $\omega_j \neq 1$, then $1+\omega+\ldots+\omega^{n-1}=0$. We can simplify our equation for the remaining eigenvalues of the system, $\lambda_j = 2+2(1+\omega_j+\ldots+\omega_j^{n-1})=2$. For any $n \in \mathbb{N}$, all eigenvalues of the system are positive, so H is positive definite. This implies that if we find a critical point, this point will be the global minimum.

Setting each component of the gradient of W equal to zero, we obtain the following system of equations:

$$\begin{bmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} m\overline{x}_\ell.$$

which has a unique solution in $x = \overline{x}_\ell \mathbf{1}_m$.

In order to meet our integer constraint and find a feasible solution to Problem (2.4), for each node $i \in \{1,\ldots,m-1\}$ we will choose $x_i = \lfloor \overline{x}_\ell \rfloor$ or $x_i = \lceil \overline{x}_\ell \rceil$. To meet our conservation of loads constraint, we solve

$$\begin{aligned} &\alpha+\beta = m, \\ &\alpha\lfloor \overline{x}_\ell \rfloor + \beta\lceil \overline{x}_\ell \rceil = m\overline{x}_\ell, \end{aligned} \tag{2.5}$$

where α and β are the number of nodes at $\lfloor \overline{x}_\ell \rfloor$ and $\lceil \overline{x}_\ell \rceil$, respectively. In the event that $\overline{x}_\ell$ is an integer, we define $\alpha = m$ and $\beta = 0$. Otherwise, it must hold that $\alpha = m(\lceil \overline{x}_\ell \rceil - \overline{x}_\ell)$, $\beta = m(\overline{x}_\ell - \lfloor \overline{x}_\ell \rfloor)$. This satisfies our other relaxed constraint $x_i \leq c_i$, $\forall\, i \in V$, because we know that $c_i \geq \lceil \overline{x}_\ell \rceil$, $\forall\, i \in V_\ell$. Permuting the set of nodes which have $x_i = \lfloor \overline{x}_\ell \rfloor$ or $x_i = \lceil \overline{x}_\ell \rceil$ has no effect on our function $W(x)$, so there can be multiple solutions. Any integer configuration for which $x_q > \lceil \overline{x}_\ell \rceil$ or $x_p < \lfloor \overline{x}_\ell \rfloor$ for some $q, p \in V_\ell$ is not a solution to Problem (2.4). With the same reasoning as in Lemma 3.1, we can achieve a reduced $W(x)$ by increasing x_p and decreasing x_q. With this, we have determined our optimal set of states. We

summarize our results with the following theorem:

Theorem 3.2. *A state x which solves Problem* (2.1) *and is therefore in the set of balanced states must satisfy $x_i = c_i$, $\forall\, i \in \tilde{V}_\ell$, $x_i = \lfloor \overline{x}_\ell \rfloor$ for α nodes in V_ℓ and $x_i = \lceil \overline{x}_\ell \rceil$ for the rest, where $\tilde{V}_\ell = \{i \in V \mid c_i \leq \lfloor \overline{x}_\ell \rfloor\}$, $\alpha = m(\lceil \overline{x}_\ell \rceil - \overline{x}_\ell)$, $V_\ell = V \setminus \tilde{V}_\ell$, $\overline{x}_\ell$ is previously defined in this section, and $m = |V_\ell|$.*

2.3.2 Main theorem

In this section we provide the main convergence theorem, under the following assumptions:

Assumption 3.3 (Graph Connectivity)**.** *The underlying undirected graph $G = (V, E)$ is connected.*

Assumption 3.4 (Capacity Distribution)**.** *For each node i with finite capacity $c_i < \max_{j \in V} x_j(0) + 1$, there is no node h with $c_h < \max_{j \in V} x_j(0) + 1$ within 3-hops of i.*

Assumption 3.5 (Neighbor Condition)**.** *If the graph is divided into subgraphs where each node i s.t. $c_i < \max_{j \in V} x_j(0) + 1$ and all neighbors of i separate the subgraphs, then each subgraph is aperiodic. Additionally, each node i with $c_i < \max_{j \in V} x_j(0) + 1$ has at least two neighbors.*

Assumption 3.6 (Initial Condition)**.** *The initial state $x(0)$ is either feasible or $\forall\, i \in V$ s.t. $x_i(0) > c_i, \forall\, \hat{\jmath} \in \operatorname{argmin}_{j \in \mathcal{N}_i} x_j(0)$, $c_j - x_j(0) \geq x_i(0) - c_i + 2$.*

Assumption 3.3 ensures interaction between all nodes, so that there is convergence towards a global average. Assumption 3.4 prevents accidental imbalancing caused from one node having to choose between two offers from two different capped nodes, sometimes leading to a load being relayed in the wrong direction. Without more information than the states and capacities of immediate neigbors, a node cannot distinguish

between accepting from the node that will result in more balancing, or accepting from the node that will create further imbalance. Assumption 3.5 ensures the system does not fail to fully converge due to periodic interactions caused by capped nodes. This assumption is crucial for being able to break down the graph and analyze its components to prove convergence over all initial conditions and many different graph sizes and configurations. For simplicity we consider a fixed underlying graph, however, Assumption 3.3 can be replaced with the assumption that, for all time k, agents are connected with others by means of graphs $G(k)$ such that $\cup_{\ell\in[k,k+B]}G(\ell)$, for some $B \geq 0$, is connected. The main effect of this will be on the convergence time. If connectivity fails during one of the phases of the algorithm, then convergence can not be guaranteed. Assumption 3.6 ensures that once an unfeasible state becomes feasible, the previous assumptions are still met. We use these assumptions to state the following theorem, the main result of this chapter:

Theorem 3.7. *Consider a network of nodes modeled by an undirected graph $G = (V, E)$, subject to the constraints listed in Problem* (2.1)*. Given Assumptions 3.3 (Graph Connectivity), 3.4 (Capacity Distribution), 3.5 (Neighbor Condition), 3.6 (Initial Condition), and applying the dynamics of our quantized distributed load balancing algorithm, any initial state $x(0)$ converges to a balanced state in finite expected time.*

2.4 Load Balancing Algorithm

This section describes our proposed load balancing algorithm that satisfies integer and capacity constraints as well as conserves all initial loads at every discrete time. We have designed a synchronous algorithm which can be divided into three main sections. In the Main Algorithm, a node runs three sub-algorithms in sequence repeatedly, and terminates when certain conditions are met. In the Offering Phase, a node selects among its neighbors with minimum load to make an offer to. If this node is at its capacity, it

may select a node with a larger load than its own to offer to, in order to maintain passing connectivity of the graph. In the Accepting Phase, a node selects among its neighbors that offered, usually selecting the node with maximal load but sometimes giving priority to a node at its capacity. In the Passing Phase, passes are made to nodes that have accepted offers, where the amount passed is calculated such that the offering node expects to minimize the difference between the two nodes' loads after the pass.

Initially define time $k = 0$, each node $i \in \{1, \ldots, n\}$ runs Main Algorithm, and the algorithm terminates when $k = k_f$, a predetermined sufficiently large time. Note, individual nodes will not be able to verify that they have converged without requiring more information than from immediate neighbors.

Algorithm 1: Main Algorithm for Node i

Inputs : $k, k_f . \forall\, j \in \mathcal{N}_i \cup i$, $x_j(k)$ and c_j.

```
while k ≤ k_f do
    Execute Offering Phase;
    Execute Accepting Phase;
    Execute Passing Phase;
    Increment k;
end
```

If all nodes' capacities are infinite, this algorithm is identical to the balancing algorithm proposed in [33], except in that algorithm the amount of load passed from i to j is $(x_i - x_j)/3$.

2.5 Convergence Analysis

We prove convergence of our algorithm to the set of balanced states by using ideas from Lyapunov stability theory; an outline of the proof is as follows. First, we define a Lyapunov function to characterize how close the system is to being balanced. This Lyapunov function also includes a term which counts how many nodes are at a

Algorithm 2: Offering Phase for Node i

Inputs : $k, k_f . \forall\, j \in \mathcal{N}_i \cup i$, $x_j(k)$ and c_j.

$x_i^{\min}(k) \leftarrow \min_{j \in \mathcal{N}_i} x_j(k)$;
$J_i(k) \leftarrow \{j \in \mathcal{N}_i \mid x_j(k) = x_i^{\min}(k)\}$;
Choose $\hat{\jmath}$ randomly from nodes in $J_i(k)$;
if $\lceil \frac{x_i(k) - x_{\hat{\jmath}}(k)}{2} \rceil > c_{\hat{\jmath}} - x_{\hat{\jmath}}(k)$ **then**
    $x_i^{\min^*}(k) \leftarrow \min_{j \in \mathcal{N}_i \setminus \hat{\jmath}} x_j(k)$;
    $J_i^*(k) \leftarrow \{j \in \mathcal{N}_i \setminus \hat{\jmath} \mid x_j(k) = x_i^{\min^*}(k)\}$;
    Choose $\hat{\jmath}^*$ randomly from nodes in $J_i^*(k)$;
    if $\frac{1}{2}(x_i(k) - x_{\hat{\jmath}^*}(k)) \geq c_{\hat{\jmath}} - x_{\hat{\jmath}}(k)$ **then**
        $\hat{\jmath} \leftarrow \hat{\jmath}^*$;
    end
end
if $x_i(k) > x_{\hat{\jmath}}(k)$ *or* $(x_i(k) \geq c_i$ *and* $k < k_f)$ **then**
    send an offer to $\hat{\jmath}$ consisting of $(i, c_i, x_i(k))$;
end

minimum value. For visualization purposes, we say that a node with this minimum value ($x_i = q$) has a q-coin or token, which is often "passed" to neighbors when a neighbor passes a load to i. We define coins, a set of rules for passing coins, and how these traverse the graph over time until they are eliminated in the proof of the theorem. Precisely, the algorithm and coin passing rules ensure that all minimum values have coins (Claim 5.1 and Claim 5.2) or are neighbors of exactly one node at its maximum capacity. Thus, coins can not be "created" by agents affected by a capacity limit, and eliminating coins guarantees minimum values necessarily increase. Claims 5.4 and 5.6 are used to bound the Lyapunov function and characterize its minimum, respectively. Claim 5.7 is used to show that the Lyapunov function is non-increasing, and then Claims 5.8, 5.9, and 5.10 combine previous results to show that the Lyapunov function is decreasing over time, and reaches its minimum in finite expected time. A finite expected time T means $\mathbf{E}[T] < \infty$. This minimum value corresponds to a balanced state, so our algorithm has converged to a desired state.

Algorithm 3: Accepting Phase for Node i

Inputs : $M_1 = \{(j_1, c_{j_1}, x_{j_1}(k)), \ldots, (j_m, c_{j_m}, x_{j_m}(k))\}$ is the set of messages received at time k.

if $|M_1| > 0$ **then**
  $x_i^{\max}(k) \leftarrow \max_{s\in\{1,\ldots,m\}} x_{j_s}(k)$;
  if $x_{j_s} > c_{j_s}$ *for any* $s \in \{1,\ldots,m\}$ **then**
    $h \leftarrow j_s$;
  else if $c_{j_s} < x_i(k)$ *for any* $s \in \{1,\ldots,m\}$ *and* $x_i^{\max}(k) \le x_i(k) + 1$ **then**
    $h \leftarrow j_s$;
  else
    $Q_i(k) \leftarrow \{j_s \in M_1 \mid x_{j_s}(k) = x_i^{\max}(k)\}$, $s \in \{1,\ldots,m\}$;
    Choose h randomly from nodes in $Q_i(k)$;
  end
  Send an acceptance to h consisting of (i);
end

Proof. Initially, if $x_i(0) > c_i$ for any node i, then the first iteration of the algorithm will proceed as follows. Any node with $x_i(0) > c_i$ will offer to its lightest neighbor j. This neighbor will accept the offer from i because it prioritizes nodes above their capacity, and because from Assumption 3.4 there can only be one neighbor with finite capacity. The amount passed from i to j will ensure that $x_i(1) = c_i$ and $x_j(1) < c_j$ due to Assumption 3.6, so after one iteration of Algorithm 1 we have a feasible load distribution that also satisfies Assumption 3.4 at $k = 1$.

Now define $z = 0$ and $q_z = \min_{i\in V}\{x_i(0) \mid x_i(0) \le c_i - 2\}$ and $Q_z = \{i \in V \mid x_i(0) = q_z,\ x_i(0) \le c_i - 2\}$. For every node j s.t. $x_j(0) = c_j$, remove one node $h \in \mathcal{N}_j$ from Q_z (if it exists), and define a q-coin at every other node in Q_z. A q-coin (or simply coin) at a node represents that this node has or can create a minimum load value. We define this notion because these values can travel around the network, and for convergence we need to track them. We will later define q-coin dynamics for all times $k \ge 0$. Over time, q-coins will be destroyed, and once there are none we will redefine new q-coins but at a greater q_z value; this will be useful in proving convergence.

Algorithm 4: Passing Phase for Node i

Inputs : $M_2 = \{(j)\}$ is the message received at time k.

if $|M_2| > 0$ **then**
  if $x_i(k) > c_i$ **then**
    $\delta \leftarrow x_i(k) - c_i$;
  else if $x_i(k) > x_j(k)$ **then**
    $\delta \leftarrow \min\left(\lceil \frac{x_i(k)-x_j(k)}{2} \rceil, c_j - x_j(k)\right)$;
  else
    $\delta \leftarrow 1$;
  end
  Pass δ load to j;
end

We consider

$$\hat{W}(x) \triangleq W(\hat{x}) = \sum_{i \in V} (\hat{x}_i - \overline{x})^2,$$

where $\overline{x} = \frac{1}{n}\mathbf{1}^\top x(0)$ is the average of the initial states and $\hat{x}$ is a modification of state x. We define $\hat{x}$ by the following rule: if for some $i \in V$, $x_i = c_i$, and $\forall\, j \in \mathcal{N}_i$, $x_j \geq x_i$, then $\hat{x}_i = x_i - 1$ and $\hat{x}_h = x_h + 1$ for some $h \in \operatorname{argmin}_{j \in \mathcal{N}_i} x_j$. Otherwise, $\hat{x}_i = x_i$. Note that the value of $\hat{W}$ is independent of the choice of h. Intuitively, $\hat{x}$ is a feasible state where all *capped* nodes (nodes in the set $\tilde{V}_\ell$) with no neighbors with smaller load have passed a load to temporarily unbalance the system; this represents a worst-case scenario in terms of minimizing W.

We define $\tilde{W}(x) = \hat{W}(x) + W_c(x)$, where $W_c(x) = (n+1)(\lceil \overline{x}_\ell \rceil - q_z) + s(x)$, $s(x) = |S(x)|$, and $S(x) = \{i \in V \mid i \text{ has a } q\text{-coin}\}$. Recall that $\overline{x}_\ell$ was found in Section 2.3.1 and corresponds to the average of the excess load that needs to be divided among the remaining non-capped nodes. We know that $0 \leq q_z \leq \lceil \overline{x}_\ell \rceil$ and $0 \leq s(x) \leq n$, so $W_c(x)$ is positive, implying $\tilde{W}(x(k))$ is positive for all $k \geq 0$. With a slight abuse of notation we denote $\tilde{W}(x(k)) = \tilde{W}(k)$, $\hat{W}(x(k)) = \hat{W}(k)$, and $s(x(k)) = s(k)$. When a q-coin is destroyed from time k to $k+1$ and if $s(k) \geq 1$ and $s(k+1) \geq 1$, then $W_c(k+1) < W_c(k)$. If $s(k+1)$

reaches zero, then at that time step we will redefine q_z and q-coins until $q_z = \lceil \overline{x}_\ell \rceil$. While $s(k)$ may increase by some amount between zero and n after redefining coins, $(n+1)(\lceil \overline{x}_\ell \rceil - q_z)$ decreases by at least $n+1$ so overall $W_c(k)$ decreases. We will later prove that $\hat{W}(k)$ is non-increasing over time, and we will use the composite $\tilde{W}(k)$ to prove convergence. Note, we cannot simply use $W(k)$ as our Lyapunov function because the behavior of nodes with finite capacities eliminates the monotonic behavior of $W(k)$.

Generally, when some node $i \in S(k)$ receives at least one load from a neighboring node j at time $k+1$, a q-coin is either passed from i to j or is destroyed. The existence of no coins on the graph implies that the graph no longer contains that particular minimum value, so that once this happens, we can increment z by one, define $q_z = q_{z-1} + 1$, and redefine coins in a similar way as done at step $k = 0$ but with respect to the new value of q_z and accounting for the possibility of a node at $q_z - 1$. This process is repeated until $q_z = \lceil \overline{x}_\ell \rceil$ or convergence to the set of nearly balanced states is achieved. We will precisely characterize the set of nearly balanced states later in order to conclude the proof.

First, we will define the motion of q-coins. Given a pass from node j to node $i \in \mathcal{N}_j \cap S(k)$ at time k, we characterize the following scenarios:

(i) Suppose that $x_i(k) = q_z$ and $x_j(k) \geq q_z + 2$. Then, the coin at i is destroyed at time $k+1$.

(ii) Suppose that $x_i(k) = q_z$ and $x_j(k) \leq q_z + 1$. Then either $x_j(k) = c_j$ and $c_j \leq q_z$, or $x_j(k) = q_z + 1$. If $x_j(k) = c_j, c_j \leq q_z$, then we must look at two cases. If there is a node $h \in \mathcal{N}_j$ s.t. $x_h(k) = x_h(k+1) = q_z$ without a coin, then the coin at i is passed to h at time $k+1$. If there is no such node h, then the coin at i is passed to j. If $x_j(k) = q_z + 1$ then we must look at more cases. If $h \in \mathcal{N}_j$ offers to j and j accepts, then $x_h(k) = c_h$ or $x_h(k) \geq q_z + 2$. If $x_h(k) = c_h$ and $c_h \leq q_z$ then the coin at i is passed to h at $k+1$. If $x_h(k) \geq q_z + 2$ or $c_h = q_z + 1$ then the coin at i is destroyed. If j receives no offers, then the coin at i is passed to j at $k+1$.

(iii) Suppose that $x_i(k) = c_i - 1$, $c_i \leq q_z$. If $x_j(k) = q_z + 1$ and there exists $h \in \mathcal{N}_j$ passing to j, then the coin at i is destroyed. If j does not have one such neighbor, then the coin at i is passed to j. If $x_j(k) \geq q_z + 2$, then the coin at i is destroyed. If $x_j(k) = q_z$ and there exists $h \in \mathcal{N}_j$ passing to j, then the coin at i is destroyed. If there is no neighbor of j offering to j then the coin at i is passed to j if j had no coin at time k, and if j did have a coin at time k then the coin at i destroyed.

(iv) Suppose that $x_i(k) = q_z - 1$, $c_i \geq q_z + 1$. If $x_j(k) = q_z + 1$ and there exists $h \in \mathcal{N}_j$ passing to j at time k, then i retains the coin. If j does not have one such neighbor, then the q-coin at i is passed to j. If $x_j(k) = c_j, c_j \leq q_z$ then the coin stays at i. If $x_j(k) = q_z + 2$ and there exists $h \in \mathcal{N}_j$ passing to j at time k, then the coin is destroyed. If there is no such neighbor, then the coin at i is passed to j. If $x_j(k) \geq q_z + 3$, then the q-coin is destroyed.

This is a complete list of possible passes involving coins following the algorithm. To prove convergence using $\tilde{W}(x)$, we first need to prove that these coins are expected to disappear in finite time if the state is not nearly balanced, and to do so we prove that a q-coin has a finite expected meeting time with a node h s.t. $x_h(k) \geq q_z + 2$. A meeting between a coin and a node h s.t. $x_h(k) \geq q_z + 2$ at time k is defined when they are neighbors. Note, the number of coins can be determined for any state $x(k)$ without knowledge of previous states, so we use both $W_c(x)$ and $W_c(k)$ interchangeably.

A few claims will help us characterize coins and the function $\hat{W}(x)$. In the rest of the claims in this chapter, we assume both Assumption 3.4 (Capacity Distribution) and Assumption 3.5 (Neighbor Condition) hold. Note, proofs for all claims are provided in the appendix.

Claim 5.1. *Any node i s.t. $x_i(k) = q_z$, $c_i \geq q_z + 2$ has a q-coin or has a neighbor j s.t. $x_j(k) = c_j$, $c_j \leq q_z$, for all $k \geq 0$.* □

Proof. Suppose coins have most recently been defined at some time $t \geq 0$, we know any node i s.t. $x_i(t) = q_z$, $c_i \geq q_z + 2$ either has a coin or a neighbor j s.t. $x_j(t) = c_j$. In the latter case, i is the only neighbor of j which does not have a coin (by how these are defined).

We examine next all of the possible ways in which a q_z value can be at a node i without a coin at some time $k \geq t$. One way for it to happen is when coins are redefined at $k \geq t$. In this way, there is $j \in \mathcal{N}_i$ s.t. $x_j(k) = c_j$ and i is the neighbor of j which does not get assigned a coin. In the next time step $k+1$, i is guaranteed to receive either a load or a coin without making a pass by case (ii) in the coin passing rules, so we see that the claim holds. Now we consider when a node i s.t. $x_i(k) = q_z$ without a coin is created by passes. In this way, it can be that (a) $x_i(k-1) = q_z - 1$ and $x_i(k) = q_z$, (b) $x_i(k-1) = x_i(k) = q_z$, and (c) $x_i(k-1) > q_z$ and $x_i(k) = q_z$.

Assume (a) for which $x_i(k-1) = q_z - 1$ and $x_i(k) = q_z$. Then, it must be that $\exists\, j \in \mathcal{N}_i$ s.t. $x_j(k-1) = c_j$. This is because a $q_z - 1$ value at i can only be created by a value of q_z at i and a neighboring agent j at a $c_j - 1$ at a time $k-2$. Since j offers to i at $k-1$, i will receive at least one load, so $x_i(k) = x_i(k-2) = q_z$ and $x_j(k) = c_j - 1$. However, this means that node i necessarily retains the coin that it had at time $k-1$.

Assume (b) for which $x_i(k-1) = x_i(k) = q_z$. Then, node i keeps the coin it has, or it must be that i passes to a j with value $x_j(k-1) = c_j - 1$ and receives from $h \in \mathcal{N}_i$, resulting in a coinless i with $j \in \mathcal{N}_i$ s.t. $x_j(k) = c_j$, $c_j \leq q_z$.

Assume (c) for which $x_i(k-1) > q_z$, and $x_i(k) = q_z$. Then, $x_i(k-1) = q_z + 1$ and i has passed a load to some j such that $x_j(k-1) = q_z$ or $x_j(k-1) = c_j - 1$. In the first case, i obtains either a coin from j, or if j had no coin then a coin from a neighbor of the capped neighbor of j, as explained in the coin passing rule (ii). In the second case, coinless i is neighbors with a capped node at its cap.

From all of the cases above, the nodes with q_z values without a coin created for the first time at k are all neighbors with a capped neighbor j at a value c_j with $c_j \leq q_z + 1$.

Due to the rules of the algorithm, we show next that at time $k+1$, this coinless q_z will disappear.

At time k, the capped neighbor j will offer back a load to i or to another neighbor h such that $x_i(k) = x_h(k) = q_z$. Note that any $h \in \mathcal{N}_j \setminus i$ such that $x_h(k+1) = q_z$ needs to have a coin. If it did not, then it would have been created close to a different capped agent at a previous time less than k, which is impossible at time $k+1$ given Assumption 5.2. Now we can have the following cases: (1) j offers back to i and i accepts, so that $x_i(k+1) = q_z + 1$ and $x_j(k+1) = c_j - 1$. (2) j offers back to i, but i accepts another offer. Then, i only accepts from a node that at least has a $q_z + 2$ load, and then $x_i(k+1) \geq q_z + 1$. (3) j offers back to another h having $x_h(k) = q_z$. It must be that h has a coin, and by the passing rules (ii), this coin is sent to i.

We see that given the first time there is a node i s.t. $x_i(k) = q_z$ without a coin, the next time step this coinless node at q_z does not exist anymore. Moreover, a coinless node at q_z is created near a neighbor at a maximum capacity that accepted a load from it. Then, the previous argument can be repeated to any time that a coinless node at q_z is created. So any node at i s.t. $x_i(k) = q_z$, $c_i \geq q_z + 2$ has an q-coin or a neighbor j s.t. $x_j(k) = c_j$ for any time k.

□

Claim 5.2. *Any node i s.t. $x_i(k) = q_z - 1$, $c_i \geq q_z + 1$ must have a q-coin, for all $k \geq 0$.* □

Proof. If a node i exists s.t. $x_i(k) = q_z - 1$ and $c_i \geq q_z + 1$, then it must be that either a coin was just defined on this node or $c_i = \infty$, $x_i(k-1) = q_z$, and $\exists\, j \in \mathcal{N}_i$ s.t. $x_j(k-1) = c_j - 1, c_j \leq q_z$. By Claim5.5, any coinless node at q_z at time $k-1$ must have a neighbor h s.t. $x_h(k-1) = c_h$, $c_h \geq q_z + 2$. By Assumption 5.2, i cannot have two capped neighbors. So i must have a coin at time $k-1$, and by the coin passing rules, this coin is retained at k, so i has at least one coin. □

Claim 5.3. *One node cannot contain two or more coins at any time k.* □

Proof. The only way a node can have two coins is if two coins are passed to the same node at one time step or a coin is passed to a node that already has a coin and it retains both. If node i receives two coins at time k then this implies that this node made two offers to neighbors, which is impossible.

A node i can contain a coin at time k only if (a) $x_i(k) = q_z$ and $c_i \geq q_z + 1$, (b) $x_i(k) = q_z - 1$ and $c_i \geq q_z + 1$, or (c) $x_i(k) = c_i - 1$ and $c_i \leq q_z$. In cases (b) and (c) node i cannot make an offer to any neighbors so i will not receive a second coin. In case (a) where $x_i(k) = q_z$ and $c_i \geq q_z + 1$, node i can only pass to a neighbor j s.t. $x_j(k) = c_j - 1$, and if this happens then the coin on j is destroyed as seen in case (iii) of coin passing rules. A coin cannot jump two or three hops to i because i already has a coin, so there is no way for a node to have two coins at any time step. □

Claim 5.4. *Any node i s.t. $c_i \geq q_z$ is lower bounded by $q_z - 1$, forward in time.* □

Proof. A node i s.t. $x_i(k) = q_z - 1$, $c_i \geq q_z$ must have a new coin defined on it at time k, must have passed to a capped node at time $k-1$, or must be a capped node with $c_i = q_z$. If coins were just redefined at time k, then there were no coins at k before the redefinition so $\exists\, j \in \mathcal{N}_i$ s.t. $x_j(k) = c_j$, $c_j \leq q_z$. If i passed to a capped node at time $k-1$ then this capped node is full at time k. If $c_i = q_z$, due to Assumption 5.2 (Capacity Distribution), there are no capped neighbors of i, so there are no nodes with lighter loads for i to pass to. We know that in each case i will receive at least one offer at time $k+1$ and again due to Assumption 5.2 (Capacity Distribution), i will not offer, so $x_i(k+1) \geq q_z$. Because any node i s.t. $c_i \geq q_z$ can only reach $q_z - 1$ under specific conditions and this value increases in one time step, it is impossible for i to reach a value less than $q_z - 1$. □

The next claim requires a precise definition of the set of nearly balanced states. By Claim 5.6, these are the states partly characterized by the minimum value of $\tilde{W}(x)$.

Intuitively, this set contains states which can be balanced in one time step and whenever possible, pairs load values at $\lfloor \overline{x}_\ell \rfloor$ with capped nodes.

Definition 5.5 (Nearly Balanced States)**.** *Consider Problem* (2.5)*. Let α be the number of nodes at $\lfloor \overline{x}_\ell \rfloor$ for some balanced state given $x(0)$ and $|\tilde{V}_\ell| = n - |V_\ell|$, where $|V_\ell|$ is the number of nodes whose capacities do not affect the solution to Problem* (2.1)*. A* nearly balanced state *satisfies one of the following conditions:*

(i) If $\alpha < |\tilde{V}_\ell|$, then there are α nodes with $x_i(k) = c_i$, $i \in \tilde{V}_\ell$ which have exactly one neighbor at $\lfloor \overline{x}_\ell \rfloor$ with the rest at $\lfloor \overline{x}_\ell \rfloor + 1$, or $x_i(k) = c_i - 1$, $i \in \tilde{V}_\ell$ which have all neighbors at $\lfloor \overline{x}_\ell \rfloor + 1$. The remaining $|\tilde{V}_\ell| - \alpha$ capped nodes are either at their cap with all neighbors at $\lfloor \overline{x}_\ell \rfloor + 1$ or below their cap by one unit with one neighbor at $\lfloor \overline{x}_\ell \rfloor + 2$ and the rest of the neighbors at $\lfloor \overline{x}_\ell \rfloor + 1$. All other nodes are at $\lfloor \overline{x}_\ell \rfloor$ or $\lceil \overline{x}_\ell \rceil$.

(ii) If $\alpha \geq |\tilde{V}_\ell|$ then each capped node below its cap by one unit has a neighbor at $\lfloor \overline{x}_\ell \rfloor + 1$ and each capped node at its cap has a neighbor at $\lfloor \overline{x}_\ell \rfloor$. All other nodes are at $\lfloor \overline{x}_\ell \rfloor$ or $\lceil \overline{x}_\ell \rceil$.

Claim 5.6. *Recall α, $\overline{x}_\ell$, and $\tilde{V}_\ell$ from Section 2.3.1, we define $\tilde{W}^* = \hat{W}^* + W_c^*$, where $W_c^* = s^*$, $s^* = \max\{0, \alpha - |\tilde{V}_\ell|\}$ and $\hat{W}^* = \hat{W}(x_{\text{best}})$ where x_{best} is a feasible minimizer of $\hat{W}$. It follows that $x(k)$ is a nearly balanced state if and only if $\tilde{W}(x(k)) = \tilde{W}^*$.* □

Proof. We first show that $W_c(k) = (n+1)(\lfloor \overline{x}_\ell \rfloor - q_z) + s(k) \geq s^*$ for all k. The first term of $W_c(k)$ is positive because, by definition of the q_z values, it holds that $q_z \leq \lfloor \overline{x}_\ell \rfloor$. Note that, when $\lfloor \overline{x}_\ell \rfloor > q_z$, then $W_c(k) \geq n + 1 > s^*$. Suppose that $\lfloor \overline{x}_\ell \rfloor = q_z$. If $s^* = 0$, then $s(k) \geq s^*$ by definition. If $s^* > 0$, then $\alpha > |\tilde{V}_\ell|$ and there are more $\lfloor \overline{x}_\ell \rfloor$ values than capped nodes to be assigned in a balanced state configuration.

The minimum number of coins at a state with $q_z = \lfloor \overline{x}_\ell \rfloor$ is s^* by the conservation of loads constraint. To see this, consider such a state for which $q_z = \lfloor \overline{x}_\ell \rfloor$. For all nodes

that are at their cap and have neighbors with coins, take one load out of the capped node and put it on its coinless neighbor. In this way, we can assume without loss of generality that the total number of coins is given by $\alpha_1 + \alpha_2$ where α_1 is the number of coins given by the q_z neighbors of nodes in $\tilde{V}_\ell$ below their cap by one unit, and α_2 is the number of $\lfloor \overline{x}_\ell \rfloor$ values which are not neighbors with any node in $\tilde{V}_\ell$. All other nodes at their cap have at most only one neighbor with a $\lfloor \overline{x}_\ell \rfloor$ value, which is coinless, and at least one other neighbor with strictly higher values. Assume the number of such capped nodes is $\gamma \leq |\tilde{V}_\ell|$. Then, the other nodes producing coins is exactly $|\tilde{V}_\ell| - \gamma$. By taking one load at a time from neighbors of capped nodes in the γ group, and respecting the q_z lower bound, we can fill up all of the capped nodes providing coins in the α_1 group. In the worst case, we create exactly γ new $\lfloor \overline{x}_\ell \rfloor$ values. By conservation of loads, by filling out all capped nodes and having $q_z = \lfloor \overline{x}_\ell \rfloor$, we necessarily obtain a balanced configuration where there are no longer higher values than $q_z + 1$. Then, the number of $\lfloor \overline{x}_\ell \rfloor$ values is exactly given by $\alpha = \alpha_1 + \alpha_2 + \gamma$. From here, $\alpha_1 + \alpha_2 = \alpha - \gamma \geq \alpha - |\tilde{V}_\ell|$. Finally, note that the minimum value s^* can always be achieved by a state with $q_z = \lfloor \overline{x}_\ell \rfloor$, and where all capped nodes are at their cap and have a neighbor at $\lfloor \overline{x}_\ell \rfloor$.

Let us now prove the claim. Suppose that $\tilde{W}(x(k)) = \tilde{W}^*$. This implies that $W_c(k) = s^*$, $\hat{W}(k) = \hat{W}^*$, and $q_z = \lfloor \overline{x}_\ell \rfloor$.

If $\alpha < |\tilde{V}_\ell|$ then $s^* = 0$, which means that there are no coins, so every node i with $x_i(k) = \lfloor \overline{x}_\ell \rfloor$, $c_i \geq \lfloor \overline{x}_\ell \rfloor + 2$ has a neighbor j s.t. $x_j(k) = c_j$, $c_j \leq q_z$. There are no nodes at $q_z - 1$ with $c_i \geq q_z + 1$ because there are no coins, and no nodes at less than $q_z - 1$ due to Claim 5.8. Every node i with $c_i \leq \lfloor \overline{x}_\ell \rfloor$ has $x_i(k) \in \{c_i, c_i - 1\}$ because $c_i \leq \lfloor \overline{x}_\ell \rfloor$. Consider all of the agents j for which $x_j(k) = c_j - 1$, $c_j \leq \lfloor \overline{x}_\ell \rfloor$. By the above, they have all neighbors at values higher or equal than $\lfloor \overline{x}_\ell \rfloor + 1 = \lceil \overline{x}_\ell \rceil$. By filling up these nodes with one load from one of the highest-loaded neighbors, we arrive at a configuration which must correspond to a balanced state, otherwise we would be able to decrease $\tilde{W}(k)$. By

the conservation of loads, there must be α values at $\lfloor \overline{x}_\ell \rfloor$ and β values at $\lfloor \overline{x}_\ell \rfloor + 1$ after this process. Then, these conditions guarantee that the state meets exactly case (i) of Definition 5.5.

If $\alpha \geq |\tilde{V}_\ell|$, then there must be exactly $s^* = \alpha - |\tilde{V}_\ell|$ coins. Since $\hat{W}(k) = \hat{W}^*(k)$, we again have that every node i with $c_i \leq \lfloor \overline{x}_\ell \rfloor$ must satisfy $x_i(k) \in \{c_i, c_i - 1\}$. Also due to $\hat{W}(k) = \hat{W}^*(k)$, all nodes are upper bounded by $\lfloor \overline{x}_\ell \rfloor + 1$. By conservation of loads we can fill up all of the capped nodes by moving sequentially one load from the highest loaded neighbors of the capped nodes. We can do this without creating any new $\lfloor \overline{x}_\ell \rfloor$ value and conserving the number of coins. This leads to a balanced state where, by conservation of loads, there are α values at $\lfloor \overline{x}_\ell \rfloor$, and only $\alpha - |\tilde{V}_\ell|$ have coins. Then, the remaining $\tilde{V}_\ell$ values at $\lfloor \overline{x}_\ell \rfloor$ must be neighbors of the $\tilde{V}_\ell$ capped nodes (one per capped node in order not to count for the number of coins). In terms of the original state, $x(k)$, it means that every node at $x_i(k) = c_i$ has a neighbor at $\lfloor \overline{x}_\ell \rfloor$ and every node at $x_i(k) = c_i - 1$ has a neighbor at $\lfloor \overline{x}_\ell \rfloor + 1$. Because the total number of loads is $\alpha \lfloor \overline{x}_\ell \rfloor + (|V_\ell| - \alpha) \lceil \overline{x}_\ell \rceil + \sum_{i \in \tilde{V}_\ell} c_i$, any coin at less than q_z means an extra load exists at another node which implies that $\hat{W}(k) > \hat{W}^*$. So there are no nodes i with $x_i(k) = \lfloor \overline{x}_\ell \rfloor - 1$, $c_i \geq \lfloor \overline{x}_\ell \rfloor + 1$. All these conditions guarantee that we are in case (ii) of Definition 5.5.

To prove necessity, we use Definition 5.5. We have already proven that s^* is the minimum, and $W_c(k) = s^*$ for any nearly balanced state. The requirements on neighbors of capped nodes as well as the rest of the nodes ensures that $\hat{W}(k) = W^*$, so any state that is nearly balanced minimizes $\tilde{W}(k)$. □

Claim 5.7. *It holds that* $\hat{W}(k) \geq \hat{W}(t) \geq W(t)$, $\forall\, t \in [k, \ldots, k_f - 1]$. □

Proof. Recall K_f as the time in which the algorithm is terminated. It holds that $\hat{W}(t) \geq W(t)$ for any t in this interval because any difference between $\hat{x}(t)$ and $x(t)$ will increase the variance, by definition of $\hat{x}(t)$. To see this, note that, in order for $\hat{x}(t) \neq x(t)$, there

must exist at least one capped node i s.t. $x_i(t) = c_i$ and $\forall\, j \in \mathcal{N}_i$, $x_j(t) \geq x_i(t)$. We can apply Equation (2), defining q as node i and p as some lightest neighbor of i. Because $x_p(t) \geq x_q(t)$, then $\Delta W = \hat{W}(t) - W(t)$ is positive, so this bound holds.

In order to prove that $\hat{W}(k) \geq \hat{W}(t)$ for all $t \in \{k, \ldots, K_f - 1\}$ in this interval, we show that $\hat{W}(t+1) \leq \hat{W}(t)$. We know that $\hat{W}(t) = \sum_{i \in V} (\hat{x}_i(k) - \overline{x})^2 = \sum_{i \in v_1} (\hat{x}_i(k) - \overline{x})^2 + \sum_{i \in v_2} (\hat{x}_i(k) - \overline{x})^2 + \ldots + \sum_{i \in v_y} (\hat{x}_i(k) - \overline{x})^2$ as long as $v_1 \cup v_2 \cup \ldots \cup v_y = V$ and $v_a \cap v_b = \emptyset, \forall\, a, b \in \{1, \ldots, y\}, a \neq b$. In the following, we find a collection of sets $v_1, v_2, \ldots, v_y$ so that their component of $\hat{W}(t)$ is nonincreasing from t to $t+1$. In doing this, recall that if a node at its cap has multiple lightest neighbors, we can select any one of them to be changed by $\hat{x}(k)$ without a loss of generality since $\hat{W}(k)$ is independent of this choice.

Because a node can pass to at most one node and receive from at most one node at time t, any group of nodes passing to and receiving from each other will form a line, where the head of the line is a node that passes and does not receive, and the tail of the line is a node that receives and does not pass (note that a ring is impossible). Initially, let us make every node in a line of passes part of a group and call it v_1, and define $\hat{W}_1(k) = \sum_{i \in v_1} (\hat{x}(k) - \overline{x})^2$. Suppose that at the tail of the line, node z is passing to j, $x_j(t+1) = c_j$, and there does not exist $h \in \mathcal{N}_j$ s.t. $x_h(t+1) < x_z(t+1)$. Then, it can be seen that $\hat{W}_1(t+1) \leq \hat{W}_1(t)$, proof available in the appendix. If $x_j(t+1) = c_j$ and there does exist $h \in \mathcal{N}_j$ s.t. $x_h(t+1) < x_z(t+1)$, then we define $\hat{x}(t+1)$ by choosing h instead of j, so we add h and any line of passes that h is in to our set v_1 and repeat the previous process with the new end node. Individually, each of the component lines of the new v_1 does not increase $\hat{W}_s(t)$ for some component s (see appendix), so their sum will not. In addition, this potential reselection made by $\hat{x}(t+1)$ at node j will ensure $\hat{W}_{s_1 \cup s_2}(t+1) < \hat{W}_{s_1 \cup s_2}(t)$, we can see this by defining $q = z$ and $p = h$ and applying Equation (2). Lastly, if $x_i(t+1) < c_i$ then we do not change v_1. After applying this process to all lines, we group the rest of the nodes into v_y, and because these nodes are not passing, receiving, or

being newly selected by $\hat{x}(t+1)$, we know $\hat{W}_y(t+1) = \hat{W}_y(t)$. We have found subsets of nodes that satisfy our exclusivity and exhaustivity conditions, and for each subset $s \in \{v_1, \ldots, v_y\}$, $\hat{W}_s(t+1) \leq \hat{W}_s(t)$, so we can conclude that $\hat{W}(t+1) \leq \hat{W}(t)$. By applying an inductive argument where $\hat{W}(t+2) \leq \hat{W}(t+1) \leq \ldots$, it is clear that $\hat{W}(k) \leq \hat{W}(t)$ for all $t+1 \leq k \leq K_f$.

□

Claim 5.8. *Given Assumptions 3.3 (Graph Connectivity), 3.4 (Capacity Distribution), 3.5 (Neighbor Condition), and 3.6 (Initial Condition), any initial state $x(0)$ will converge to the set of nearly balanced states in finite expected time.*

Proof. As discussed early in this section, in one time step an infeasible state becomes a feasible state. Because of Claim 5.7, we can use $\tilde{W}(k) = \hat{W}(k) + W_c(k)$ as a Lyapunov function for this system, for $k \geq 1$. We treat $W_c(k)$ as the main component and use $\hat{W}(k)$ to indicate balancing for some cases where $W_c(k)$ does not capture it. The function $\tilde{W}(k)$ is positive, non-increasing under the dynamics, and any decrease has a fixed lower bound. Proving that $\tilde{W}(k)$ will decrease in finite expected time from any state $x(k)$ that is not nearly balanced is enough to prove convergence to the set of nearly balanced states.

First, we look at the case where $\forall\, i \in V, c_i > \max_{i \in V} x_i(0)$, which is the case where the capacities of the nodes never influence the load balancing process. In this case, $\hat{W}(k) = W(k)$ and we can just use W_c as our Lyapunov function, because when $W_c(k) = W_c^*$ and there are no capped nodes, we satisfy (ii) of Definition 5.5. If $W_c(k) > W_c^*$ then at least one coin needs to be destroyed to reach the nearly balanced set. We start by assuming there is a coin at node i. We know that any node at q_z contains a coin, this is true because capacities do not influence balancing in this case. If $\exists\, j \in \mathcal{N}_i$ s.t. $x_j(k) \geq q_z + 2$, then j will offer to i and $W_c(k)$ will decrease in one time step. In the event that all neighbors of the coin for some time interval are at $q_z + 1$ and $W_c(k)$ remains constant

for this interval, then we have a homogeneous Markov chain over this time interval. We know this is irreducible (because the graph is connected) and recurrent (because the probability of the coin leaving and never returning to a node is 0), so with probability 1 the coin has a finite hitting time to any node on the graph. For the expected hitting time to be finite, we require positive recurrence, which is satisfied because null recurrence is impossible for a discrete time finite state space Markov Chain. The motion of the coin and any value that the coin can balance with (so at least at $q_z + 2$) are independent, so a finite expected hitting time induces a finite expected meeting time, and then we know $W_c(k)$ will decrease.

Now suppose there are multiple coins in the graph with no capped nodes and node i has a coin. Note that any neighbor j of i s.t. $x_j(k) \geq q_z + 2$ will offer to i or to another node at q_z. Whichever node it offers to, this node will accept from j or from a neighbor at least at $q_z + 2$, so a coin is guaranteed to be destroyed. For there to be a time interval where no coins are destroyed, there cannot be any neighbors of coins at $q_z + 2$ or higher. Since we only allow neighbors of a node with a coin to be at q_z or $q_z + 1$, the coin has a positive probability of being passed to all nodes at $q_z + 1$, and all nodes at q_z have coins. To find the expected hitting time of any coin to some node j, we locate the closest coin to j that is d-hops away. The probability of a coin reaching j in d time steps is at least as probable as if there were only this one coin, because any other coins offering to nodes along the path to j will increase the likelihood of the coin reaching j, and because we are looking at the path from the closest coin, there is no negative interference from other coins. This means we have a finite upper bound on the expected hitting time, implying finite expected meeting time and decrease of $W_c(k)$.

Once all q-coins have been destroyed at time k, we increment z, define $q_z = q_{z-1} + 1$, define $Q_z = \{i \in V \mid x_i(0) \in \{q_z, q_z - 1\},\ x_i(0) \leq c_i - 2\}$, and for every node j s.t. $x_j(0) = c_j$, remove one node $h \in \mathcal{N}_j$ from Q_z (if it exists), and define a q-coin at every

other node in Q_z. Define $S(k) = \{i \in V \mid i \text{ has a } q\text{-coin at time } k\}$, $s(k) = |S(k)|$, and iterate. An increase in q_z means we have tightened the bounds on certain nodes' loads because of Claim 5.4. There cannot be a node i s.t. $c_i \geq q_z$, $x_i(k) = q_z - 2$ via Claim 5.4, so our definitions still hold. Coins are destroyed and redefined until $W_c(k) = W_c^*$, and the state is nearly balanced.

We now consider the case of the graph containing at least one node h s.t. $c_h \leq \max_{i \in V} x_i(0)$. If $\tilde{W}(k) > \tilde{W}^*$, then we are not at a nearly balanced state. So we will first assume that we have exactly one coin that needs to be destroyed, then there must also exist at least one node with a large load (either at least at $q_z + 2$ without a neighbor at $c_j - 1$ or at least at $q_z + 3$ and with a neighbor at $c_j - 1$).

We will show that from any initial position, there exists a sequence of states that can occur with some positive probability to make the coin meet with the larger node. This is sufficient to ensure that there is a finite expected meeting time between the two. We will assume there is exactly one coin and one large node, as if there are multiple then the meeting time between any two is less.

First we must see that because capped nodes have at least two neighbors, they must choose between them to make offers or accept offers. If a capped node is full and both of its neighbors are equal, then it chooses randomly between them to offer to. If its neighbors are not equal, then if offers to the lowest of them. In the next time step, given no change to the state of the other neighbors, it will either accept from the larger of the two or choose randomly if they are equal. This randomness is important in order to be able to break the graph into subgraphs and analyze them as a whole, as seen in Section 2.8.3. Periodic behavior of loads across nodes is possible if the state is already nearly balanced, but if it is not, then there must be a large node somewhere or a coin that can be destroyed.

We will break G into subgraphs, defined by removing each capped node and its

edges. If a coin lies on one of the capped nodes, then it can be treated as if it were in any of its neighboring subgraphs. Suppose that the coin on node i and a large load on node h remain in the same subgraph during an infinite number of iterations, and we would like these to meet in order to move towards the set of nearly balanced states. Without interference from capped nodes, there is a finite expected meeting time between the coin and the large load. Capped nodes interfere with the motion of coins when the coin stays within one hop of a capped node j over time. This occurs when $x_i(k) = q_z$, $x_j(k) = c_j - 1$, and whenever $x_i(k) = q_z - 1$, all neighbors of i are at q_z or less. However, this condition will not hold because the large node is in this subgraph, creating enough load difference to remove periodicity. While the passes to and from a capped node can be periodic, when both of these values are in the same subgraph, this periodicity is broken by an immediate decrease in $\tilde{W}(k)$ (for example if the subgraph is small), or by the graph itself being aperiodic due to Assumption 3.5 which leads to a finite expected meeting time.

If the coin and large load are not in the same subgraph, then there must be one or more capped nodes between them. Note that the difference between the coin's load and the large load is at least two. All neighbors of the coin and of the large load must be within one, otherwise $\tilde{W}(k)$ would decrease. So somewhere between these two values, there exists a capped node with values between q_z and $q_z + 1$ on one side and $q_z + 1$ and $q_z + 2$ on the other side (or a potentially larger difference). Because each separating subgraph is aperiodic (via Assumption 3.5), the algorithm shows sufficient random behavior to ensure than at one time step this capped node can pass to the lighter of the two sides, and then receive from the heavier of the two sides, completing a relay of a load across the capped node, full proof found in Section 2.8.3. If this relay does not reduce $\tilde{W}(k)$ directly, then we now have a scenario with potential balancing on a subgraph (similar to the above case), which will reduce $\tilde{W}(k)$ in finite expected time. Note that by the strong connectivity assumption on G, the relay of loads across capped nodes ensures that, if the

coin and the higher load were not eliminated, there will have to be an infinite number of times for which they reach the same subgraph, a contradiction with the previously discussed case.

This means that any coin will have a finite expected meeting time with a node with large load, and eventually $W_c(k)$ converges to W_c^*. We also require $\hat{W}(k) = \hat{W}^*$. If $\hat{W}(k) > \hat{W}^*$ while $W_c(k) = W_c^*$ then there exists a large load. With a proof similar to the previous coin traversal argument, this large load will also traverse the graph until it meets a node at $\lfloor \overline{x}_\ell \rfloor$ with positive probability, so we again have finite expected meeting time of these values. For all cases we have a finite number of finite expected times in which $\tilde{W}(k)$ decreases by an amount with a nonzero lower bound, and $\tilde{W}(k)$ is lower bounded by $\tilde{W}^*$, so we can conclude the result.

□

Claim 5.9. *The set of nearly balanced states is invariant under the dynamics of our algorithm.*

Proof. Recall the definition of the set of nearly balanced states from Definition 5.5. If $\alpha < |\tilde{V}_\ell|$, then capped nodes can pass to neighbors at $\lfloor \overline{x}_\ell \rfloor$ or $\lfloor \overline{x}_\ell \rfloor + 1$ and nodes at $\lfloor \overline{x}_\ell \rfloor + 1$ can pass to nodes at $\lfloor \overline{x}_\ell \rfloor$ or $c_i - 1$. If $\alpha \geq |\tilde{V}_\ell|$, then capped nodes can pass to neighbors at $\lfloor \overline{x}_\ell \rfloor$ and nodes at $\lfloor \overline{x}_\ell \rfloor + 1$ can pass to nodes at $\lfloor \overline{x}_\ell \rfloor$ or $c_i - 1$. All of these passes result in a state which satisfies the constraints on the set of nearly balanced states. □

Claim 5.10. *If $x(k)$ is a nearly balanced state for some $k < k_f$, then $x(k_f)$ is a balanced state.*

Proof. We require $x(k)$ to be a nearly balanced state for some $k < k_f$. Due to Claim 5.9 we know $x(k_f - 1)$ is also nearly balanced. The last iteration of the algorithm ensures that for our last iteration k_f, if $x_i(k_f - 1) < c_i \leq \lfloor \overline{x}_\ell \rfloor$, then $x_i(k_f) = c_i$ and the load was received from a node at $\lfloor \overline{x}_\ell \rfloor$ or $\lfloor \overline{x}_\ell \rfloor + 1$. No node i at its capacity is permitted to offer to

a node j s.t. $x_j(k_f - 1) \geq x_i(k_f - 1)$. In conclusion, our final state is: $\forall\, i \in \tilde{V}_\ell$, $x_i(k_f) = c_i$, and $\forall\, i \in V_\ell$, $x_i(k_f) \in \{\lfloor \overline{x}_\ell \rfloor, \lceil \overline{x}_\ell \rceil\}$. □

□

2.6 Simulations

2.6.1 Example Problem

We have simulated this algorithm on a wide range of graphs, each with various capacity configurations and initial load conditions. One sufficiently complex graph is seen in Figure 2.1. The square vertices represent nodes with finite capacities and the circular vertices represent nodes with infinite capacities. We can see that some nodes with finite capacities are located at central points on the graph, and if removed will disconnect the graph. The initial load configuration is shown in Figure 2.2 and the final load configuration is shown in Figure 2.3, after k_f iterations of our algorithm. In both of these figures, finite capacity values are shown as triangles. We can see in Figure 2.4 the non-increasing behavior of $\tilde{W}(k)$ and convergence to the optimal value.

2.6.2 Convergence Rate

We have generated three types of graphs of varying sizes to estimate the time complexity of our algorithm as a function of the number of nodes in a graph. The basis for each graph is a line graph, with small capacities placed every 4 nodes. With some probability, we generate edges for every other pair of non-capped nodes. This probability is zero for type one, it is simply a line graph. This probability is $16/N^2$ for type two, so that the average number of edges added stays constant as the size of the graph grows. This probability is 0.5 for type three, so the connectivity of the graphs scales with the

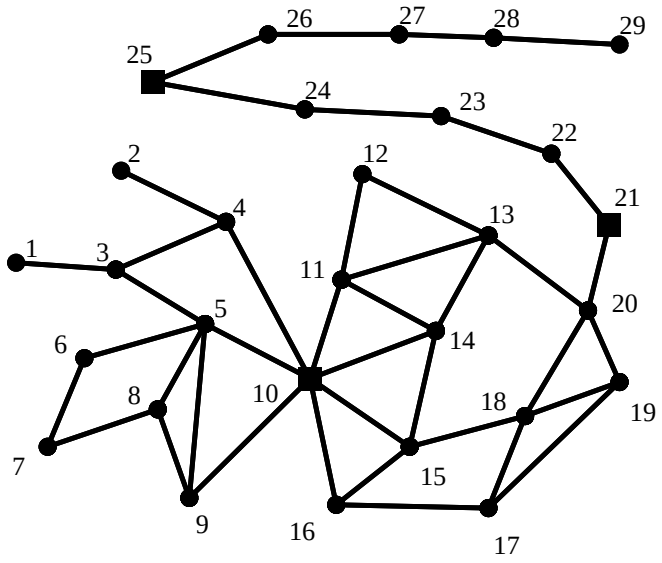


Figure 2.1: Example connected graph

number of nodes.

In Figure 2.5, the average convergence time over 100 trials is plotted for each graph type, and the 20% confidence interval is shown (this small interval is chosen to see the increase in error as the number of nodes grows, it is cut off in Figure 2.5 for graph type 1 with 32 nodes and graph type 2 with 64 nodes). We note the superlinear relationship between the number of nodes and the average convergence time for each graph type. The confidence interval is very large for a large number of nodes, due to the increased effect of randomness and the lower probability of a decrease in the Lyapunov function when the state approaches nearly balanced.

We have also simulated a modified version of the algorithm incorporating minimum load requirements at certain nodes, for example $x_i(k) \geq b_i, \forall k$ for some node i. A node at its minimum cannot make an offer, a node chooses which neighbor to accept from based on the how large the offer is (not their load), and the amount passed from a node cannot create a state where $x_i(k) < b(i)$ for any node i. This extension seems to work well, though convergence for all configurations of capacities and load limits has not been

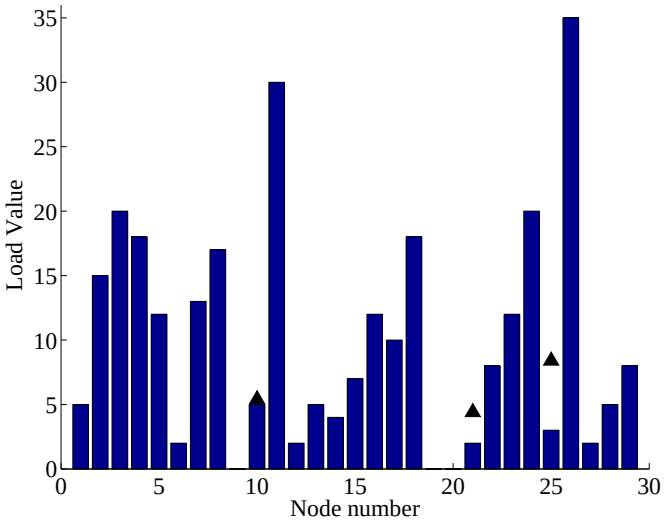


Figure 2.2: Initial load distribution

analyzed. A new Lyapunov function should be found to characterize the evolution of the system, and convergence can likely be proven under some conditions on the distribution of both nodes with finite capacities and nodes with minimum load requirements.

2.7 Conclusions and Future Work

We have designed a new quantized distributed load balancing algorithm that accounts for maximum node capacities. We proved convergence to a set of balanced states in expected finite time under some assumptions on the location as well as a neighbor condition of nodes with small capacities. We tested this algorithm over a variety of graphs and initial conditions, verifying its convergence properties and time complexity. Simulations also showed good performance of the algorithm extended to include nodes with a minimum capacity along with nodes with maximum capacity. We will aim to fully characterize these cases in future work as well as investigate upper bounds on the rate of convergence of the algorithm.

Chapter 2, in part, has been published in proceedings of IEEE Int. Conf. on

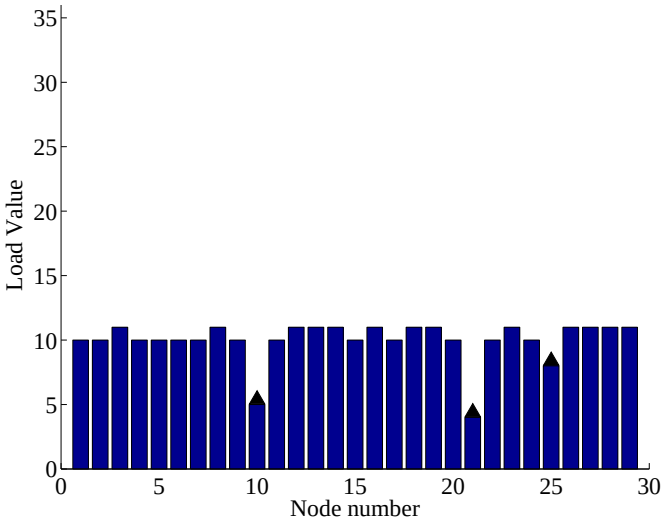


Figure 2.3: Final load distribution

Decision and Control, Los Angeles, CA, USA, December 2014, by E. Gravelle and S. Martínez, and in IEEE Transactions on Control of Networked Systems, 4(2):279–287, 2017, by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

2.8 Additional Proofs

The following three proofs are used in the main theorem result in this chapter.

2.8.1 Proof that lines of passes without capped nodes always cause a decrease in W

Consider

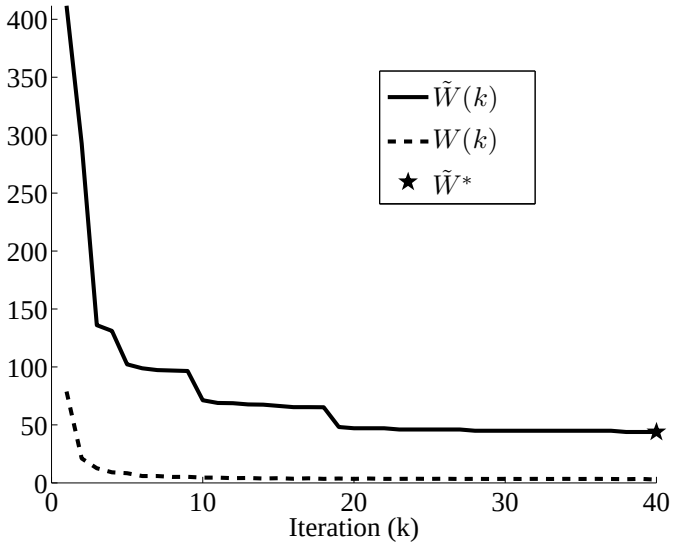


Figure 2.4: Plot of $W(x)$ over k_f iterations

$$W(k) = \sum_{i=1}^{n} (x_i(k) - \overline{x})^2 = (x_a(k) - \overline{x})^2 + \ldots + (x_z(k) - \overline{x})^2$$

$$W(k+1) = (x_a(k+1) - \overline{x})^2 + \ldots + (x_z(k+1) - \overline{x})^2.$$

We would like to prove $\Delta W \leq 0$. Compute,

$$\Delta W = W(k+1) - W(k) = (x_a(k+1)^2 - 2x_a(k+1)\overline{x} + \overline{x}^2 + \ldots + x_z(k+1)^2$$
$$- 2x_z(k+1)\overline{x} + \overline{x}^2) - (x_a(k)^2 - 2x_a(k)\overline{x} + \overline{x}^2 + \ldots + x_z(k)^2 - 2x_z(k+1)\overline{x} + \overline{x}^2).$$

Simplifying,

$$\Delta W = x_a(k+1)^2 + \ldots + x_z(k+1)^2 - x_a(k+1)^2 - \ldots -$$
$$x_z(k+1)^2 - 2\overline{x}(x_a(k+1) - x_a(k) + \ldots + x_z(k+1) - x_z(k)).$$

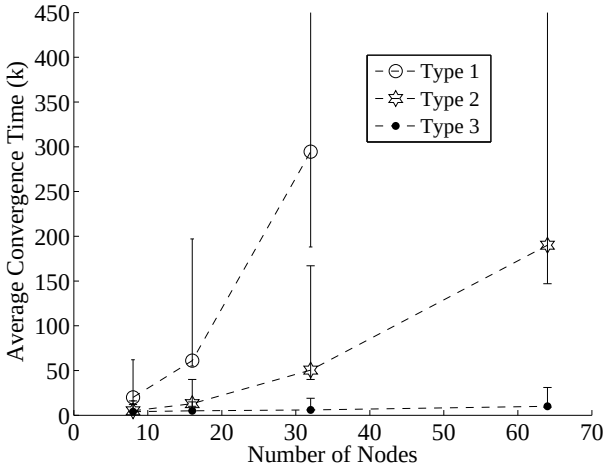


Figure 2.5: Plot of average convergence time

Since we know $\overline{x}$ is constant over time,

$$\frac{1}{n}\sum_{i=1}^{n} x_i(k) = \frac{1}{n}\sum_{i=1}^{n} x_i(k+1),$$

and substituting,

$$\Delta W = x_a(k+1)^2 + \ldots + x_z(k+1)^2 - x_a(k+1)^2 - \ldots - x_z(k+1)^2 - 2\overline{x}\left[\sum_{i=1}^{n} x_i(k+1) - \sum_{i=1}^{n} x_i(k)\right].$$

The update rule is $x_i(k+1) = x_i(k) + \delta_{\text{in}} - \delta_{\text{out}}$, so

$$\Delta W = (x_a(k) - \delta_{\text{ab}})^2 + (x_b(k) + \delta_{\text{ab}} - \delta_{\text{bc}})^2 + \ldots + \left(x_y(k) + \delta_{\text{xy}} - \delta_{\text{yz}}\right)^2 + \left(x_z(k) + \delta_{\text{yz}}\right)^2 - x_a(k)^2 - \ldots - x_z(k)^2.$$

Further simplifying,

$$\Delta W = -2\delta_{\text{ab}}x_a + \delta_{\text{ab}}^2 + 2(\delta_{\text{ab}} - \delta_{\text{bc}})x_b + (\delta_{\text{ab}} - \delta_{\text{bc}})^2 + \ldots + (\delta_{\text{xy}} - \delta_{\text{yz}})x_y + (\delta_{\text{xy}} - \delta_{\text{yz}})^2 + 2\delta_{\text{yz}}x_z + \delta_{\text{yz}}^2.$$

We break this into two components,

$$\Delta W = \Delta W_1 + \Delta W_2,$$

where

$$\Delta W_1 = 2\delta_{\text{ab}}\left[\delta_{\text{ab}} - (x_a - x_b)\right] + \ldots + 2\delta_{\text{yz}}\left[\delta_{\text{yz}} - (x_y - x_z)\right],$$

and

$$\Delta W_2 = -2\left[\delta_{\text{ab}}\delta_{\text{bc}} + \ldots + \delta_{\text{xy}}\delta_{\text{yz}}\right].$$

We know $\delta_{\text{ab}} = \dfrac{x_a - x_b}{2}$ or $\delta_{\text{ab}} = \dfrac{x_a - x_b + 1}{2}$. Simplifying the first term of ΔW_1, it can be either $\dfrac{-(x_a - x_b)^2}{2}$ or $-\dfrac{1}{2}(x_a - x_b + 1)(x_a - x_b - 1)$ depending on the effect of rounding. This guarantees that $\Delta W_1 \leq 0$ because a requirement of a pass to occur is $x_a - x_b \geq 1$, and this can be extended to every term in ΔW_1. The end result is that $\Delta W_1 \leq 0$.

Every term in ΔW_2 is strictly negative by definition of the line graph, so as long there are at least 3 nodes involved in this line, we have $\Delta W_2 < 0$. From these two statements, it is clear that $\Delta W \leq 0$ for a line with 2 nodes and $\Delta W < 0$ for a line with at least 3 nodes. We can see that as long as loads are conserved along the line and $\overline{x}$ stays constant over k, it has no effect on $\Delta W(k)$.

2.8.2 Proof that lines of passes including maximum capacities always cause a decrease in $\hat{W}$

We look at a line of passes, and define i as the node passing without receiving and j as the node receiving without passing. In order for $x(k) \neq \hat{x}(k)$ or $x(k+1) \neq \hat{x}(k+1)$, we have $x_i(k) = c_i$ or $x_j(k+1) = c_j$. We will only look at $\hat{W}$ for this line, with node denoted $\{i, a, b, \ldots, y, z, j\}$.

We know that

$$\hat{W}_\ell(x(k)) = W_\ell(\hat{x}(k)) = \sum_{p \in \ell} (\hat{x}_p(k) - \overline{x})^2,$$

and we would like to prove that

$$\hat{W}_\ell(x(k+1)) \leq \hat{W}_\ell(x(k)), \forall k \in \mathbb{N}.$$

Let's assume that $x_i(k) = c_i$ and $x_j(k+1) = c_j$ as a worst case. We know that $\hat{x}_i(k) = x_i(k) - 1$, $\hat{x}_a(k) = x_a(k) + 1$, $\hat{x}_z(k+1) = x_z(k+1) + 1$, and $\hat{x}_j(k+1) = x_j(k+1) - 1$. We must show

$$\sum_{p \in \ell} (\hat{x}_p(k+1) - \overline{x})^2 \leq \sum_{p \in \ell} (\hat{x}_p(k) - \overline{x})^2,$$

or

$$\begin{aligned}
&\hat{x}_i(k+1)^2 - 2\hat{x}_i(k+1)\overline{x} + \overline{x}^2 + \hat{x}_a(k+1)^2 - 2\hat{x}_a(k+1)\overline{x} + \overline{x}^2 + \ldots + \\
&\qquad \hat{x}_z(k+1)^2 - 2\hat{x}_z(k+1)\overline{x} + \overline{x}^2 + \hat{x}_j(k+1)^2 - 2\hat{x}_j(k)\overline{x} + \overline{x}^2 \leq \\
&\qquad\qquad \hat{x}_i(k)^2 - 2\hat{x}_i(k)\overline{x} + \overline{x}^2 + \hat{x}_a(k)^2 - 2\hat{x}_a(k)\overline{x} + \overline{x}^2 + \ldots + \\
&\qquad\qquad\qquad \hat{x}_z(k)^2 - 2\hat{x}_z(k)\overline{x} + \overline{x}^2 + \hat{x}_j(k+1)^2 - 2\hat{x}_j(k)\overline{x} + \overline{x}^2.
\end{aligned}$$

We know that

$$\begin{aligned} x_i(k+1) &= x_i(k) - \delta_{ia} \\ x_a(k+1) &= x_a(k) + \delta_{ia} - \delta_{ab} \\ &\vdots \\ x_z(k+1) &= x_z(k) + \delta_{yz} - \delta_{zj} \\ x_j(k+1) &= x_j(k) + \delta_{zj}, \end{aligned}$$

and

$$\begin{aligned} \hat{x}_b(k) = x_b(k), \ldots, \hat{x}_z(k) = x_z(k), \hat{x}_j(k) = x_j(k), \\ \hat{x}_i(k+1) = x_i(k+1), \hat{x}_a(k+1) = x_a(k+1), \ldots, \hat{x}_y(k+1) = x_y(k+1), \end{aligned}$$

because nodes that both pass and receive cannot be at their cap. So we substitute in to get

$$\begin{aligned} &x_i(k+1)^2 - 2x_i(k+1)\overline{x} + \overline{x}^2 + x_a(k+1)^2 - 2x_a(k+1)\overline{x} + \overline{x}^2 + \ldots + \\ &(x_z(k+1)+1)^2 - 2(x_z(k+1)+1)\overline{x} + \overline{x}^2 + (x_j(k+1)-1)^2 - 2(x_j(k+1)-1)\overline{x} + \overline{x}^2 \leq \\ &(x_i(k)-1)^2 - 2(x_i(k)-1)\overline{x} + \overline{x}^2 + (x_a(k)+1)^2 - 2(x_a(k)+1)\overline{x} + \overline{x}^2 + \ldots + \\ &x_z(k)^2 - 2x_z(k)\overline{x} + \overline{x}^2 + x_j(k)^2 - 2x_j(k)\overline{x} + \overline{x}^2. \end{aligned}$$

We can now substitute further,

$$(x_i(k)-\delta_{ia})^2-2(x_i(k)-\delta_{ia})\overline{x}+\overline{x}^2+(x_a(k)+\delta_{ia}-\delta_{ab})^2-2(x_a(k)+\delta_{ia}-\delta_{ab})\overline{x}+\overline{x}^2+\ldots+$$
$$(x_z(k)+\delta_{yz}-\delta_{zj}+1)^2-2(x_z(k)+\delta_{yz}-\delta_{zj}+1)\overline{x}+\overline{x}^2+(x_j(k)+\delta_{zj}-1)^2-$$
$$2(x_j(k)+\delta_{zj}-1)\overline{x}+\overline{x}^2\leq(x_i(k)-1)^2-2(x_i(k)-1)\overline{x}+\overline{x}^2+$$
$$(x_a(k)+1)^2-2(x_a(k)+1)\overline{x}+\overline{x}^2+\ldots+x_z(k)^2-2x_z(k)\overline{x}+\overline{x}^2+x_j(k)^2-2x_j(k)\overline{x}+\overline{x}^2,$$

and expand to get

$$x_i(k)^2-2\delta_{ia}x_i(k)+\delta_{ia}^2-2(x_i(k)-\delta_{ia})\overline{x}+\overline{x}^2+$$
$$x_a(k)^2+2(\delta_{ia}-\delta_{ab})x_a(k)+(\delta_{ia}-\delta_{ab})^2-$$
$$2(x_a(k)+\delta_{ia}-\delta_{ab})\overline{x}+\overline{x}^2+\ldots+x_z(k)^2+2(\delta_{yz}-\delta_{zj}+1)x_z(k)+$$
$$(\delta_{yz}-\delta_{zj}+1)^2-2(x_z(k)+\delta_{yz}-\delta_{zj}+1)\overline{x}+\overline{x}^2+x_j(k)^2+$$
$$2(\delta_{zj}-1)x_j(k)+(\delta_{zj}-1)^2-2(x_j(k)+\delta_{zj}-1)\overline{x}+\overline{x}^2\leq$$
$$x_i(k)^2-2x_i(k)+1-2(x_i(k)-1)\overline{x}+\overline{x}^2+x_a(k)^2+2x_a(k)+1-$$
$$2(x_a(k)+1)\overline{x}+\overline{x}^2+\ldots+x_z(k)^2-2x_z(k)\overline{x}+\overline{x}^2+x_j(k)^2-2x_j(k)\overline{x}+\overline{x}^2.$$

Canceling some terms we are left with

$$-2\delta_{ia}x_i(k)+\delta_{ia}^2+2(\delta_{ia})\overline{x}+2(\delta_{ia}-\delta_{ab})x_a(k)+(\delta_{ia}-\delta_{ab})^2-2(\delta_{ia}-\delta_{ab})\overline{x}+\ldots+$$
$$2(\delta_{yz}-(\delta_{zj}-1))x_z(k)+(\delta_{yz}-(\delta_{zj}-1))^2-2(\delta_{yz}-(\delta_{zj}-1))\overline{x}+$$
$$2(\delta_{zj}-1)x_j(k)+(\delta_{zj}-1)^2-2(\delta_{zj}-1)\overline{x}\leq 2(x_a(k)-x_i(k)+1).$$

So now

$$2\delta_{ia}(x_a(k) - x_i(k) + \delta_{ia}) + 2\delta_{ab}(x_b(k) - x_a(k) + \delta_{ab}) + \ldots + 2\delta_{yz}(x_z(k) - x_y(k) + \delta_{yz}) + 2(\delta_{zj} - 1)(x_j(k) - x_z(k) + (\delta_{jz} - 1) - 2(\delta_{ia}\delta_{ab} + \ldots + \delta_{yz}(\delta_{zj} - 1)) \leq 2(x_a(k) - x_i(k) + 1),$$

and because we know $\delta_{ia} = 1$ for the case where the first node is capped,

$$2\delta_{ab}(x_b(k) - x_a(k) + \delta_{ab}) + \ldots + 2\delta_{yz}(x_z(k) - x_y(k) + \delta_{yz}) + 2(\delta_{zj} - 1)(x_j(k) - x_z(k) + (\delta_{jz} - 1) - 2(\delta_{ia}\delta_{ab} + \ldots + \delta_{yz}(\delta_{zj} - 1)) \leq 0.$$

For all $\delta_{ab}, \ldots, \delta_{yz}$, we know $\delta_{\text{fg}} \leq \frac{x_f(k) - x_g(k)}{2}$ or $\delta_{\text{fg}} \leq \frac{x_f(k) - x_g(k) + 1}{2}$, so these terms simplify to either $-2\delta_{fg}^2$ or $-2\delta_{fg}(\delta_{fg} - 1)$ which are never positive given $\delta_{fg} \geq 1$. We know that $1 \leq \delta_{zj} \leq \frac{x_z(k) - x_j(k) + 1}{2}$, which means that the last term is upper bounded by $-2\delta_{zj}(\delta_{zj} - 1)$ which is also never positive, so $\hat{W}_\ell(x(k+1)) \leq \hat{W}_\ell(x(k))$. A similar argument can be made for the cases where $x_i(k) = c_i$, $x_j(k+1) < c_j$ and $x_i(k) < c_i$, $x_j(k+1) = c_j$.

2.8.3 Proof of destruction of coin in finite time

Define G_c, a condensed version of G as follows. Nodes of G_c:

- Labeled as $B_i = \{i\} \cup \mathcal{N}_i$, where $c_i \leq \max_{i \in V} x_i(0) + 1$.
- Remove $G \setminus \bigcup_i B_i$ and define G_r as the remaining connected graphs. G_r is a node in G_c.

Edges of G_c:

- (B_i, G_r) is an edge $\iff$ there exists a node h in B_i and a node ℓ in G_r s.t. $(h, \ell) \in E$.

Assumptions:

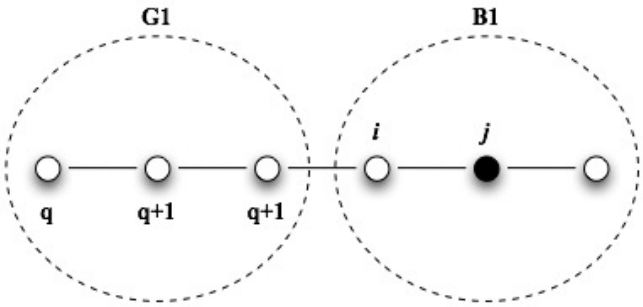
- G_c is strongly connected.
- (B_i, B_j) is never directly connected, whereas (B_i, G_r) can be.
- Each G_r is aperiodic.

Suppose G is not balanced, then coins must move to meet nodes at values greater than $q+1$.

Proof outline:

Coins are at nodes with $x_i \in \{q, q-1, c_i\}$. If q and $\geq q+2$ often come to a G_r and never balance with other values, it means they never meet with values far from their own. By the same arguments as when there are no capped nodes, they will meet in finite expected time. Thus, nodes at q or $\geq q+2$ must be confined into subsets of G_c separated by at least one B_i.

Assume first all q values are in G_1. If these never balance, then all other values at G_1 are q or $q+1$.If there were no $q+1$ values in G_1, since G_1 is connected to a B_1, a q always reaches B_1. If there are $q+1$ values at G_1, and no coins are destroyed, then there will be load transfers that result in passing of coins in G_1. The $q+1$ values in G_1 are not forced by B nodes because they are at least one hop apart.



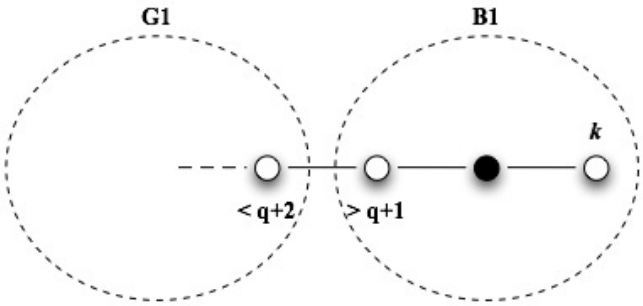
- If $x_i \geq q+2$ with $x_j = c_j$, this will result into a value $\geq q+2$ inside g_1 which is impossible.

- If $x_i \geq q+2$ with $x_j = c_j - 1$ with $x_j = c_j - 1$, it does not affect G_1, so q^* can approach n.
- If $x_i = q-1$, i will never accept from n.
- If $x_i \in \{q, q+1\}$, i will never accept from n.

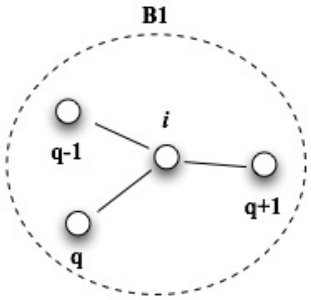
In all cases, q^* will meet with B with non-zero probability. Since G_1 is aperiodic, the times at which this happens are not periodic. So the only way in which q^* can be confined in some parts of G_c is by blocking B sections. This can occur if B does not accept or B does not release q^* values.

Suppose B_1 persistently interacts with q^* in G_1 (by strong connectivity of G_c there has to be such a B_1 interacting with q^* if coins are not destroyed, else coins would be traveling back to a B_1 from some other side). These time are never periodic.

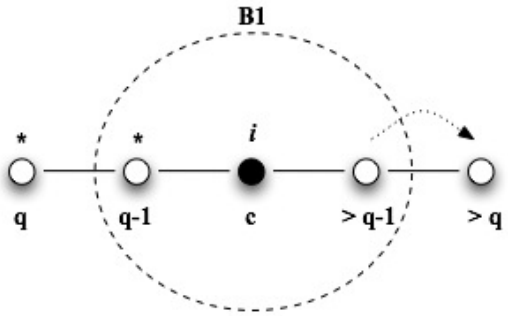
Suppose coins are not destroyed by the interaction with B_1. Then the only way q^* can meet with B_1 is via $q-1, q$, and $q+1$ values else eventually this will be the case or the coin destroyed: if $x_k \geq q+2$ then with some probability a value at $\geq q+2$ will move into G_1. If $x_k = q$ or $x_k = q+1$ then B_1 will balance across c to have $q, q+1$ everywhere.



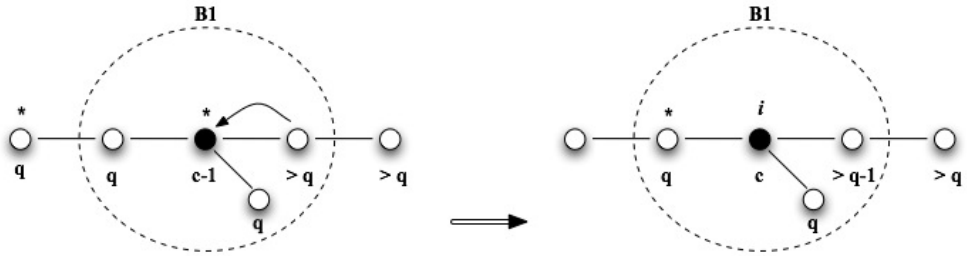
In B_1 we can separate nodes into groups, because i will choose randomly amongst nodes with the same value, we can treat the whole group as one node.



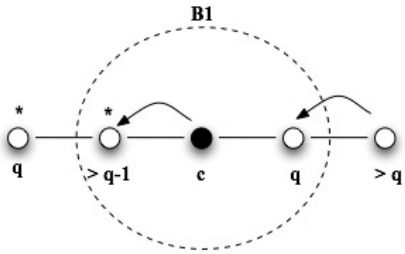
Case 0. Assume there exists $q-1$ value in B. By assumption B is blocking, so these values are $\geq q+1$. Then we have



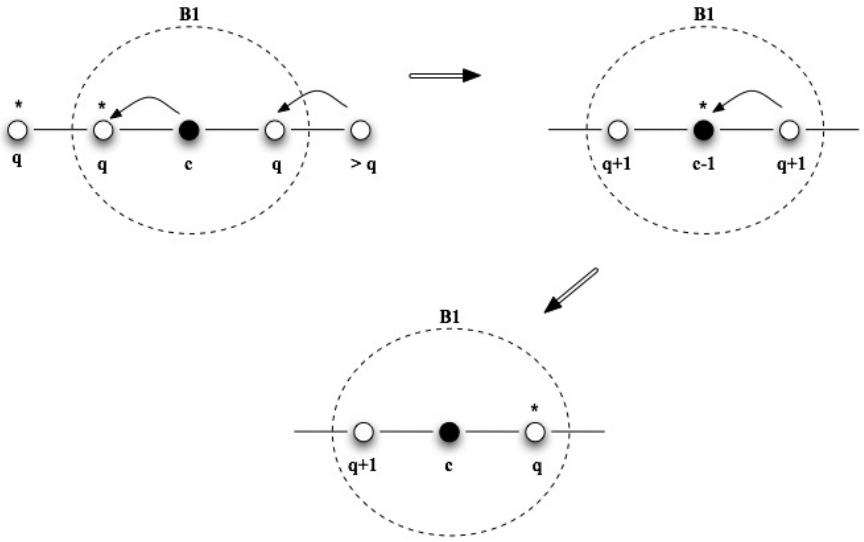
In this case i transfers to $q-1$ deterministically. With a nonzero probability, we get to



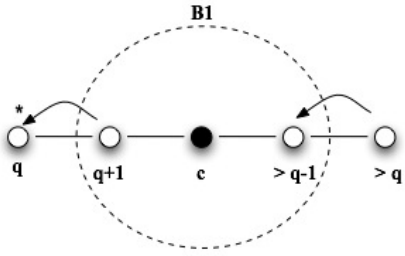
q^* on the left persists interacting with B, and will interact at a configuration as follows:



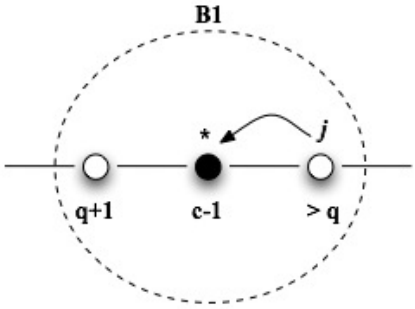
Case 1.



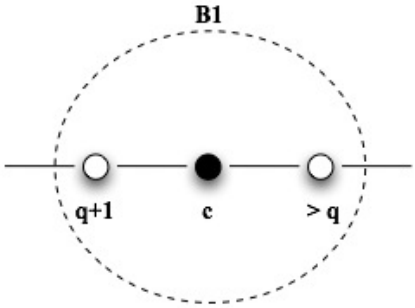
From Case 1, we arrive at Case 2:



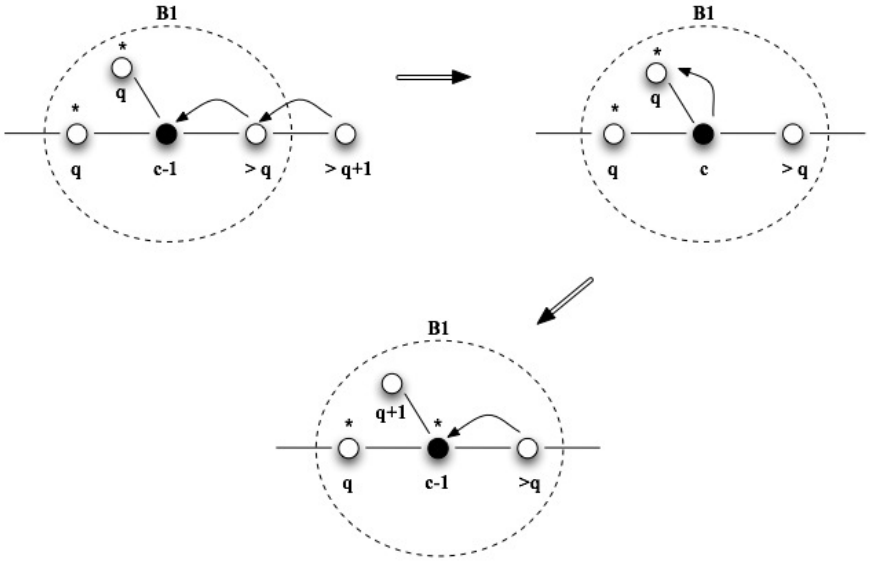
Then we have two subcases, Case 2.1 is



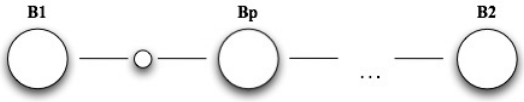
If $x_j = q + 1$ we have a pass with non-zero probability resulting in:



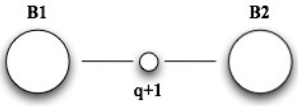
Case 2.2 is



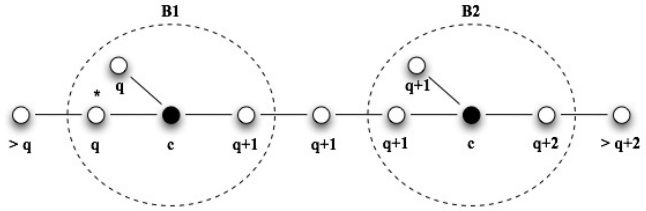
So having only one B blocking is not possible. We could have more than one B blocking, let these be B_1 and B_2, this looks like



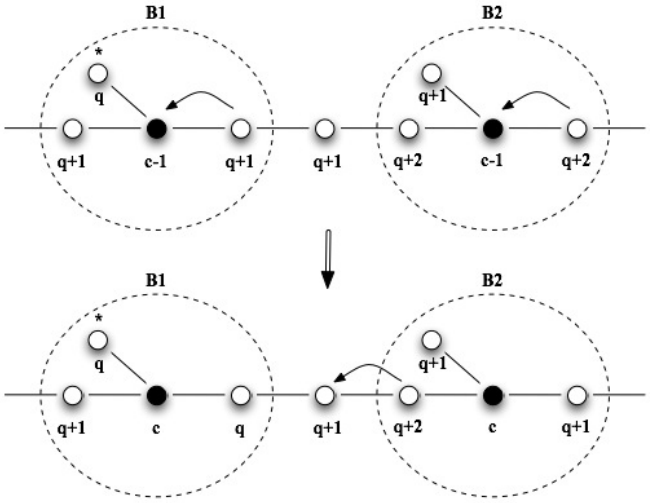
If coins are not destroyed then $q+2$ values always exist, and we can reduce the above situation to



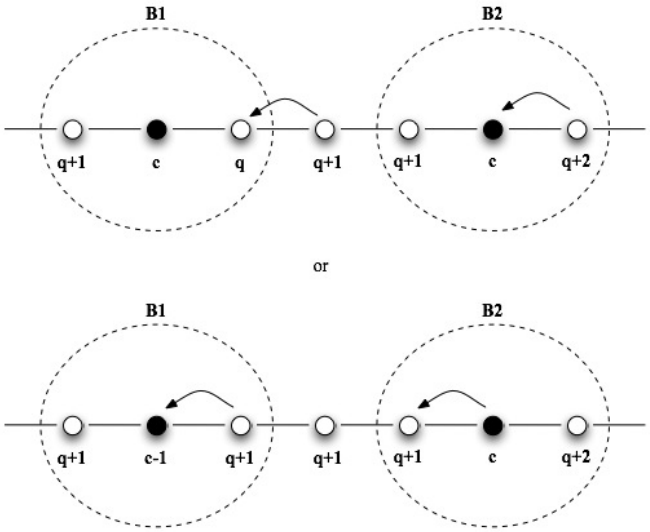
In all, we have



Then with a non-zero probability we get to



Other cases are



in which with a non-zero probability we fall under Case 2.2.

Chapter 3

Distributed Dynamic Lane Reversal and Rerouting for Traffic Delay Reduction

3.1 Abstract

Traffic congestion is a major source of delays in modern road networks. Motivated by this, we propose two distributed algorithms to reduce delays: a dynamic lane reversal algorithm and a rerouting algorithm. When there is a density imbalance on a road, time can be saved by reallocating lanes from the less dense side to the more dense side, which motivates dynamic lane reversal. When a road has greater density than nearby roads, time can be saved by redirecting flow into the least congested roads, this motivates dynamic rerouting. Given a communication system between infrastructure and vehicles on the road, the local state of the network can be approximated and utilized by the algorithms to minimize travel time. In order to provide a better fundamental understanding of the system dynamics, we analyze equilibrium conditions for the system and prove

convergence of the lane reversal algorithm to a critical point. Overall performance is also examined in simulation.

3.2 Introduction

Congestion is a major source of traffic delays in modern road networks, but the problem can be mitigated by smarter traffic systems. Significant imbalances of traffic density in a given road network can arise due to many events, such as when there is a large flow of vehicles towards an industrial center in the morning, a large event ends and there is a mass of flow out from large event, or there is an accident which creates heavy congestion on one side of a road. Modern infrastructure endowed with new information technology requires no additional space or construction and can substantially reduce overall traffic delays. Motivated by this, here we investigate the implementation and benefits of lane reversal and traffic rerouting distributed algorithms that can improve traffic flow.

In particular, recent advances in design, performance, and cost of autonomous vehicles (see [8]) has fueled a growing interest in Autonomous Intersection Management (AIM), an efficient policy for coordinating autonomous vehicles using an intersection manager (IM) to safely pass through an intersection [15]. With the help of the AIM policy and vehicle-to-infrastructure communications, an approximation of the state of traffic can be constructed. The IM can then implement more dynamic procedures to reverse one or more lanes or communicate a new route to some vehicles if traffic delays will be reduced. The future presence of autonomous vehicles is also important in implementing the actual lane reversal and vehicle rerouting, as physically moving a barrier to reverse a lane is a slow process that can take hours, [14], yet merely indicating a lane's direction or a new route for a vehicle is likely to cause driver confusion and increase risk of accident. With

advances in vehicle autonomy, lane reversal and rerouting are less restricted by physical safety considerations and can be achieved through simple communication from the traffic signal to the vehicle.

Many recent papers have furthered Autonomous Intersection Management. Batch processing of reservations in AIM to enforce liveness is proposed in [2]. An auction-based scheme under AIM is analyzed in [10]. Local information is shared and utilized to minimize delay time under Greenshield's traffic model in [48]. Some effort has also recently been put towards solving vehicle routing problems in modern context. A provably safe distributed solution for coordinating vehicles outside an intersection is provided in [42]. Work has also been done analyzing traffic evolution over networks. Classical traffic models are examined in a network setting in [47]. Passivity is used to generalize the network flow control problem in [45]. A solution to the problem of assigning freight loads to available carriers given unbalanced network conditions is found in [1].

Much of the literature concerning lane reversal discusses evacuation procedures in order to respond effectively to natural disasters, [11], [44]. These papers propose the solution of lane reversal to accommodate emergency evacuation in a non-dynamic way. Some works discuss procedures and results for location-specific cases where lane reversal would be beneficial, [50], [49]. More recently, some have attempted to further improve results through dynamic lane reversal. The solution presented in [23] requires a centralized computer to find an allocation strategy, with a minimum timestep of one hour. In [32], the authors formulate a model and present a centralized solution which does not use network dynamics. In [22], dynamic lane reversal is implemented in limited form on a single road and tested in simulation.

3.2.1 Contributions

In this chapter we extend the cell transmission model to characterize the evolution of vehicle density in a road network and the effect of both lane reversal and rerouting on these dynamics. We establish objective functions with the goal of minimizing total vehicle time spent on the road, and propose two algorithms. Using lane reversal, we propose a distributed *dynamic* algorithm to efficiently calculate and implement an appropriate lane allocation and prove convergence of the algorithm to a more efficient solution. An important aspect of this work is to provide a fundamental understanding of the system dynamics. To do so, we analyze the long term behavior of a road network with balanced lanes, and establish its convergence to an equilibrium under certain regularity conditions on its sources and sinks. We also propose a distributed rerouting algorithm to more efficiently achieve this long-term equilibrium. We show through simulations performance gains using lane reversal and rerouting on various initial conditions.

3.2.2 Outline

In Section 3.3, we define our objective and formulate two problems under different constraints to be solved. In Section 3.4, we provide stability results of lane reversal, and we present the lane reversal algorithm. In Section 3.5, we provide analysis of weight-balanced traffic networks and we present the rerouting algorithm. In Section 3.6, we simulate and discuss both algorithms to further verify their effectiveness. In Section 3.7, we conclude our work and discuss future possible extensions.

3.3 Problem Statement

We consider traffic evolving over a road network. Each *road* consists of one or two *sides* for each direction of traffic flow and which have a given number of lanes.

In addition, each side is divided into *cells* of length L, which are used to describe the evolution of traffic density, see Figure 3.1.

We define a directed graph $G_C = (C, E_C)$ of cells $i \in C$, such that $(j,h) \in E_C$ if traffic can flow from cell j to cell h. A side is defined as the set of connected cells bounded by a source, sink, or an intersection manager (IM). A source (resp. sink) is a special cell in which traffic only flows out (resp. flows in), while an IM is an intelligent traffic management system at an intersection of roads. The set of all roads is denoted by R and the set of neighbors of road r is denoted by $\mathcal{N}_r$, where two roads are neighbors if they share an intersection. We denote S as the set of all sides, $p, -p \in S$ are the two sides of a road, and $n = |C|$. The intersection graph $G_Z = (Z, E_Z)$ consists of the vertex set Z containing all IMs and edges $(z_1, z_2) \in Z$ if there is exactly one road connecting intersections z_1 and $z_2 \in Z$. A cell which flows into a sink is contained in set $\underline{B}$ and a cell which receives flow from a source is contained in set $\overline{B}$.

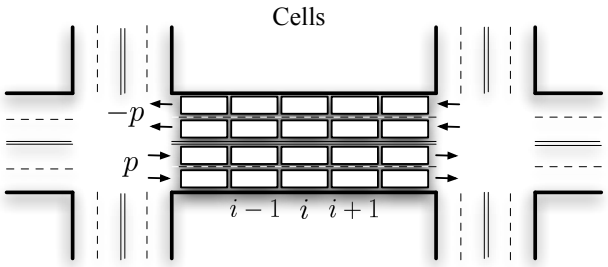


Figure 3.1: Road divided into cells

3.3.1 Traffic Model

The following traffic model is based on the *Lighthill-Whitham-Richards Partial Differential Equation*, [30] and [40], to describe the evolution of vehicle density $\rho \in \mathbb{R}$ on each side,

$$\partial_t \rho + \partial_x Q(\rho) = 0. \tag{3.1}$$

This equation maintains conservation of mass, and the flow function $Q(\rho)$ is given by

$$Q(\rho) = \begin{cases} v_f \rho, & \rho \leq \rho_c, \\ \frac{v_f \rho_c}{\rho_{\text{jam}} - \rho_c}(\rho_{\text{jam}} - \rho), & \rho > \rho_c, \end{cases} \tag{3.2}$$

where $\rho \leq \rho_c$ is the condition for free flow, $\rho > \rho_c$ is the condition for congested flow, v_f is the free flow speed of the vehicles, ρ_{jam} is the density at which a traffic jam occurs, and ρ_c is the critical density value where maximum flow occurs, see Figure 3.2. This model is based on experimental data and is commonly used to model traffic flow, particularly because it is a simple model that captures the wave behavior of traffic.

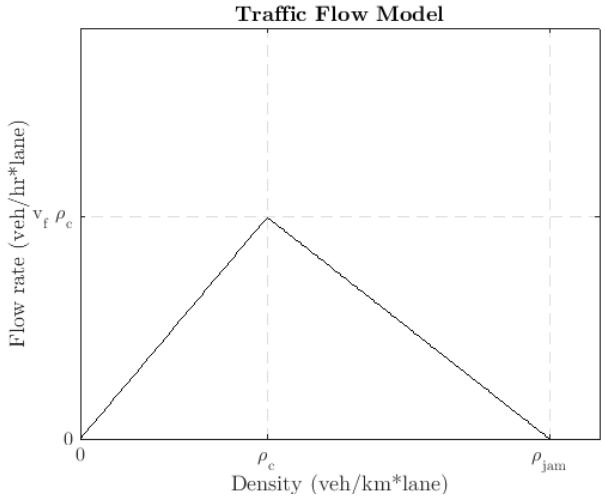


Figure 3.2: Vehicle flow model

The *Cell Transmission Model* [12] is a discretization of (3.1) using time step Δt and spatial step Δx, where is assumed that all cells have length $L = \Delta x$. For convenience, k indexes the discrete time step, with $t = t_0 + k\Delta t$. For a cell i with exactly one in-neighbor $i-1$ and one out-neighbor $i+1$, the density of the cell is updated according to

$$\rho_i(k+1) = \rho_i(k) + \frac{\Delta t}{L(\ell_p + u_p)}(q_{i-1,i}(k) - q_{i,i+1}(k)),$$

where i is contained on side p of road r, ℓ_p is the number of default lanes of side p, $\rho_i(k)$ is the density (veh/lane-km) of vehicles on i at time k, $q_{a,b}$ is the flow rate (veh/hr) from cell a to cell b, and $u_p(k) \in \{1-\ell_p, \ldots, \ell_{-p}-1\}$ is the number of additional lanes on side p. The constraint $u_p + u_{-p} = \ell_p + \ell_{-p}$ must hold to keep the total number of lanes in a road constant, where $u_{-p} \in \{\ell_{-p}-1, \ldots, \ell_p - 1\}$ is the number of additional lanes on side $-p$. We have

$$q_{i-1,i}(k) = \min\{\mathbf{q}_{i-1}(k), \mathbf{q}_i(k)\}, \tag{3.3}$$

with the piecewise functions $\mathbf{q}_{i-1}(k)$ and $\mathbf{q}_i(k)$ defined as

$$\mathbf{q}_{i-1}(k) = \begin{cases} v_f(\ell_p + u_p(k))\rho_{i-1}(k), & \rho_{i-1}(k) \le \rho_c, \\ v_f(\ell_p + u_p(k))\rho_c, & \rho_{i-1}(k) > \rho_c, \end{cases}$$

$$\mathbf{q}_i(k) = \begin{cases} v_f(\ell_p + u_p(k))\rho_c, & \rho_i(k) \le \rho_c, \\ \frac{v_f\rho_c}{\rho_{\text{jam}}-\rho_c}(\ell_p + u_p(k))(\rho_{\text{jam}} - \rho_i(k)), & \rho_i(k) > \rho_c. \end{cases}$$

Intuitively, the flow from $i-1$ to i is restricted when $\rho_{i-1}(k)$ is small or $\rho_i(k)$ is large [7].

Cells can also be connected to sources or sinks of various strengths, these make up the boundary to the system. A source or sink is just like another cell but with an *effective density* given by

$$\rho_i = \alpha_i\rho_c, \qquad i \text{ is a source},$$

$$\rho_i = (1-\beta_i)\rho_c, \quad i \text{ is a sink},$$

where $\alpha_i(\beta_i)$ is the strength of the source (sink), resp.

To model a network of roads at intersections, the flow out of a cell must equal the sum of flows into other cells. We define a matrix $K = \{k_{ij}\} \in \mathbb{R}^{n\times n}$ where k_{ij} contains the fraction of vehicles which move from cell i to cell j. For now, we assume K is constant. If j is the only out-neighbor of i in G_C then $k_{ij} = 1$, but if j is one of multiple out-neighbors, then $k_{ij} < 1$. The flow out of any cell i, based on conservation of mass, is given by

$$q_i^{\text{out}}(k) = \sum_{j\in\mathcal{N}_i^{\text{out}}} k_{ij} q_{i,j}(k), \tag{3.4}$$

where $\mathcal{N}_i^{\text{out}}$ is the set of out-neighbors of i in G_C. In this model, intersections are assumed to be small compared to the length of each cell, so the time spent in the intersection is negligible. The role of an efficient Autonomous Intersection Management policy is important in this assumption.

We similarly define

$$q_i^{\text{in}}(k) = \sum_{h\in\mathcal{N}_i^{\text{in}}} k_{hi} q_{h,i}(k), \tag{3.5}$$

where $\mathcal{N}_i^{\text{in}}$ is the set of in-neighbors of i in G_C.

The evolution of any cell in the network is given by

$$\rho_i(k+1) = \rho_i(k) + \frac{\Delta t}{L(\ell_p + u_p)}(q_i^{\text{in}}(k) - q_i^{\text{out}}(k)), \quad i \in C. \tag{3.6}$$

To enable lane reversal, the control input $u \in \mathbb{Z}^{|R|}$ determines the number of lanes per road which directly affects that road's density, see Figures 3.3 and 3.4.

Based on conservation of mass, cell i on side p is updated after lane reversal as follows:

$$\rho_i(k^+) = \rho_i(k) \cdot \frac{\ell_p + u_p(k)}{\ell_p + u_p(k^+)},$$

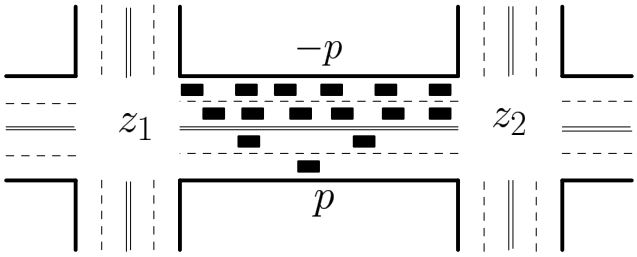


Figure 3.3: Vehicles before lane reversal

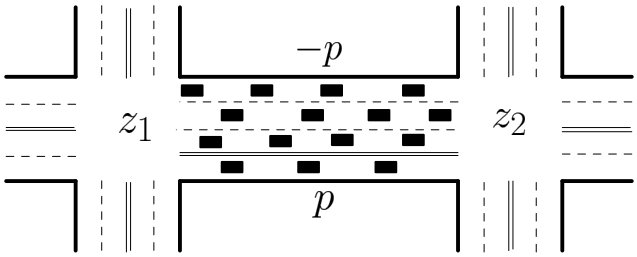


Figure 3.4: Vehicles after lane reversal

where u_p is the control before the update. For analysis purposes we assume that the change in road density is instantaneous, based on an assumption that vehicles respond quickly to a lane opening or closing. The clearing time t_c, the time it takes for all vehicles to vacate a lane being reversed, is also assumed to be zero. Lane clearing can realistically be performed in 15 seconds or less under most traffic conditions in which a lane clearing occurs, so this assumption is reasonable, [22].

3.3.2 Problem Formulation

To characterize performance of the system, we define the objective function as the time spent of each vehicle in the system G_{C} summed over every vehicle, or

$$\overline{W}(u) = \sum_{w=1}^{N} \left(t_w^{\ell} - t_w^{e} \right),$$

where t_w^ℓ is the time in which vehicle w leaves G through a sink, t_w^e is the time in which vehicle w enters G through a source, and N is the total number of vehicles that spent time within the system. The total time spent is inversely proportional to the total flow rate, so total time can be approximated as

$$\begin{aligned}\overline{W}(u) &\approx \frac{N}{q_{\text{avg}} \sum_{p \in S} \ell_p}, \\ &\approx \sum_{k=0}^{k_f} \left(\frac{N}{\sum_{i \in C} (q_i^{\text{in}}(k) + q_i^{\text{out}}(k))/2} \right),\end{aligned}$$

where q_{avg} is the average flow rate in G. The average of flow in and out of each cell is required to account for coupled dynamics.

We define the first control input as the directional lane allocation of each road $u \in \mathbb{Z}^{|R|}$, where $u_p = 1$ corresponds to reversing one lane from the default lanes in the direction of $-p$ to the direction of p in the road $r \in R$. The goal is to minimize $\overline{W}(u)$ while satisfying two physical constraints, one which maintains the total number of lanes of a roadway (the sum of lanes in both directions is constant), and the other which requires a positive integer number of lanes. This is stated as

Problem 1:

$$\begin{aligned}\underset{u \in \mathbb{R}^{|R|}}{\text{maximize}} \quad & W(u) = \sum_{k=0}^{k_f} \left(\sum_{i \in C} (q_i^{\text{in}}(k) + q_i^{\text{out}}(k)) \right) \\ \text{subject to} \quad & u_p \in \{-\ell_p + 1, \ldots, \ell_{-p} - 1\}, \\ & u_{-p} \in \{-\ell_{-p} + 1, \ldots, \ell_p - 1\}, \\ & u_p + u_{-p} = \ell_p + \ell_{-p}, \forall\, p, -p \in S.\end{aligned}$$

A point u^* is a critical point for Problem 1 if u^* satisfies the above constraints and if $W(u^*) \geq W(u)$ for all u s.t. $\forall\, p \in S, u$ satisfies the above constraints and $u_\zeta^* = u_\zeta$, $\forall\, \zeta \neq p$.

If vehicles can be redirected through intersections, then $K = \{k_{ij}\} \in \mathbb{R}^{n \times n}$ is the

control variable, where k_{ij} is the proportion of vehicles flowing from cell i to cell j. Each non-zero value is lower bounded by a value $k_{\min}$ in order to maintain connectedness of the graph. Assuming a uniform critical density value in the network and given boundary conditions, maximum flow is obtained by distributing as much flow as possible into uncongested lanes. This can be formulated as

Problem 2:

$$\begin{aligned} \underset{K\in\mathbb{R}^{n\times n}}{\text{minimize}} \quad & \tilde{W}(K) = \sum_{i\in C} \max\{0, \rho_i(t+1) - \rho_c\} \\ \text{subject to} \quad & (K\mathbf{1}_n)_i = 1, \forall\, i \notin \underline{B}, \\ & (K\mathbf{1}_n)_i = 0, \forall\, i \in \underline{B}, \\ & K \in \text{sparse}(G_{\text{C}}), \\ & k_{ij} \in [k_{\min}, 1], \forall\, (i,j) \in E. \end{aligned}$$

3.3.3 Approximation of the State

Here, we will use the assumptions employed in [22] for an intersection manager (IM) to approximate the state of the traffic on the roads at the intersection. Vehicles have unique identifiers and transmit a message within $D \approx 300$ meters to the IM for a reservation request to cross intersections more efficiently. The IM at intersection $z \in Z$ maintains a counter variable $\overline{z}_p$ for road side p, adding one to $\overline{z}_p$ when it receives a notification message from a vehicle on road side p and subtracting one from $\overline{z}_p$ whenever a vehicle from road side p with a confirmed reservation is expected to leave the road and enter the intersection. The state of road side p at time t is calculated as

$$\rho_p(k) = \frac{\overline{z}_p(k)}{(\ell_p + u_p)\min\{L, D\}}. \tag{3.7}$$

If $L > D$ then assume that the state of the entire road is equal to the state in the nearest section. Note, with more sensing than just at intersections, the state of the roads can be more accurately approximated, so smaller cells can be used. This approximation is used in both algorithms to determine whether or not travel efficiency can be improved.

3.4 Lane Reversal Policy

In this section we provide a distributed Lane Reversal Algorithm together with its stability properties. The performance of the algorithm is also analyzed in Section 3.6.

3.4.1 Lane Reversal Algorithm

Problem 1 is a non-convex, non-smooth optimizal control problem with integer constraints. We assume that there is an intersection manager z_1 and z_2 at both ends of each road, and that z_1 is assigned its control. This IM requires estimates of the road states from its neighboring IMs and from the neighbors of z_2 to construct the complete local state. These estimates are calculated by counting vehicles in and out of each road as explained in Section 3.3.3 and in Equation (3.7).

We define $C_p = \{a \in C \mid a \text{ is a cell of } p\}$ for $p \in S$, and similarly $C_r = C_p \cup C_{-p}$, where $p, -p$ are the sides of road $r \in R$. Suppose that $T(t') \in \{1, \dots, \overline{T}\}$ represents a clock ticking from 1 to $\overline{T}$ at each IM synchronously, where $\Delta t' << \Delta t$ is a smaller discrete time step. Each IM updates its assigned roads on specific ticks, which are given by a schedule $\Lambda(r) \in \{1, \dots, \overline{T}\}$, computed during an initialization phase. As an example, in the road network in Figure 3.5 each road with the same number Λ can update simultaneously. By means of the flag function "`to_update`," computations are reduced to cases when changes in the neighboring conditions can lead to non-trivial updates. When the turn of an IM to update takes place (line 5), then, in order to find the best control policy for some

road r with sides $p, -p \in S$ while keeping other roads fixed, $W_r(u + \omega_r \Delta_r)$ is maximized over $\omega_r \in \Omega_r$ in the Lane Reversal Algorithm. Here,

$$W_r = \sum_{k}^{k+\eta} \sum_{a \in C_{r \cup N_r}} (q_a^{\text{in}}(k) + q_a^{\text{out}}(k)),$$

where η is the width of the optimization window, note that for large η, accurate prediction of the local state could require more states than just immediate neighbors. In addition, $\Omega_r = \{-\ell_p + 1, \ldots, \ell_{-p} - 1\}^\eta$ is any sequence of controls and $\Delta_r \in \mathbb{Z}^n$ has zeros everywhere except 1 for each component $i \in C_p$ and -1 for each component $j \in C_{-p}$. Since in real road networks most roads have 4 or less lanes, an exhaustive search is computationally inexpensive in this domain. If a trivial update takes place, then a new update for neighboring roads is not necessary. This is encoded by setting the `to_update` function equal to zero, otherwise this function is set equal to one. State estimates are updated and information on updated controls, states and the `to_update` function is communicated to neighbors. The algorithm runs until time k_f.

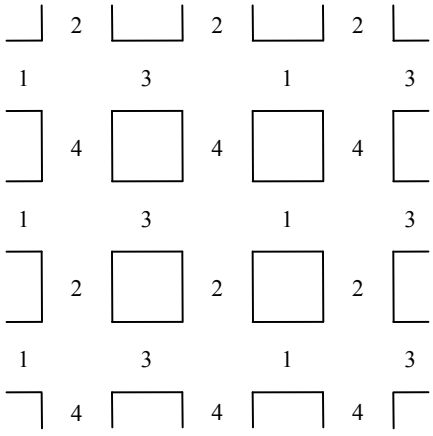


Figure 3.5: Schedule of road network

Algorithm 5: Lane Reversal Algorithm of IM z

Initialize time $t' = 0$, schedule $\Lambda(r)$, $\forall\, r \in R$;
Initialize to_update$(r) = 1$, $\forall\, r \in R$;
for *all* $r \in R$ *with sides* $p, -p$ *controlled by* z *and* t' **do**
    Update $u_{p'}$, $u_{-p'}$, to_update(r'), and $\rho_{i'}$ if messages were received from neighbors;
    if to_update $= 1$ *and* $\Lambda(r) = T(t')$ **then**
        $\omega_r^* \leftarrow \text{argmin}_{\omega_r \in \Omega_r} W_r(u + \omega_r \Delta_r)$;
        $u_\nu^+ \leftarrow u_\nu$, $\forall\, \nu \in R \setminus \{p, -p\}$;
        $u_p^+ \leftarrow u_p + \omega_r^*$;
        $u_{-p}^+ \leftarrow u_p - \omega_r^*$;
        if $u_p^+ = u_p$ **then**
            to_update$(r) \leftarrow 0$;
        else
            to_update$(\lambda) \leftarrow 1$, $\forall\, \lambda \in \mathcal{N}_r$;
            to_update$(r) \leftarrow 0$;
        end
        Initiate lane swap, set $\rho_i^+ = \rho_i \cdot \frac{\ell_r + u_r}{\ell_r + u_r^+}$, $\forall\, i \in C_p$ and $\rho_j^+ = \rho_j \cdot \frac{\ell_r - u_r^+}{\ell_r - u_r}$, $\forall\, j \in C_{-p}$;
        Transmit u_p^+, u_{-p}^+, to_update$(\lambda) \forall\, \lambda \in \mathcal{N}_r$, ρ_i^+ $\forall\, i \in C_p$, and ρ_i^+ $\forall\, i \in C_{-p}$ values to neighbors of r;
    end
    $t' \leftarrow t' + \Delta t'$;
end

3.4.2 Stability Analysis of Lane Reversal

We first establish an upper bound on the objective function:

Lemma 4.1. *Under the constraints given in Problem 1, the objective function satisfies*

$$W(u) \leq 2 v_f \rho_c \ell n,$$

at each time step, assuming that $\ell_p = \ell$, $\forall\, p \in S$. *This upper bound is achieved when roads are lane-balanced (for any path on the network, the number of lanes remains constant) and* $\rho_i = \rho_c, \forall\, i \in C$. *Intuitively, this is the state when flow through each lane is maximized*

over the whole network, and there is no congestion formed through lane merging. □

The proof for this lemma is omitted, as it is simply calculated by maximizing each cell's flow.

Lemma 4.2. *The* Lane Reversal Algorithm *converges in finite time to a critical point* u^* *of Problem 1 under the listed constraints.* □

Proof. The update u^+ is implemented in the Lane Reversal Algorithm by evaluating $W_r(u)$ and choosing ω_r which maximizes this value. Note that the algorithm constrains u_p^+ s.t. $u_p^+ \in \{1-\ell_p, \ldots, \ell_{-p}-1\}$. Since a local maximizer of $W_r(u)$ also maximizes $W(u)$ and the algorithm maintains a schedule which is compatible with the separability of W (no two road neighbors update simultaneously), it is guaranteed that $W(u^+) \geq W(u)$. In this way, W is a monotonically non-decreasing function through the algorithm. Using a discrete-time Lyapunov stability argument with W, asymptotic convergence to a point u^* satisfying the constraints of Problem 1 for which W can not be improved by modifying u^* entry-wise can be guaranteed. Due to the finite discrete state space, convergence occurs in finite time.

□

3.5 Vehicle Rerouting Policy

In this section, we provide a stability analysis for the road density evolution under the dynamics (3.3) to (3.6), under the assumption of balanced sources and sinks. This motivates the distributed rerouting algorithm, which is simulated in Section 3.6. For this section only, we assume that there is no lane reversal occuring, and that the number of lanes on each road are equal, so $\ell_p + u_p = \ell_{p'} + u_{p'} = \ell, \forall\, p, p' \in S$.

3.5.1 Stability Analysis of Weight-Balanced Traffic Networks

Because one solution to Problem 2 is achieved when as many vehicles are in free flow as possible, we are interested in finding conditions that guarantee that system achieves this state naturally. Such fundamental analysis helps gain intuition and a better understanding of how the system behaves. We find one such sufficient condition here.

We define $\alpha = [\alpha_1, \ldots, \alpha_n]^\top$, where $\alpha_i \in [0, 1]$ is the strength of the source connected to cell i ($\alpha_i = 0$ if no source is connected to cell i). Similarly, $\beta = [\beta_1, \ldots, \beta_n]^\top$ where $\beta_i \in [0, 1]$ is the strength of the sink connected to cell i ($\beta_i = 0$ if no sink is connected to cell i). We use two assumptions:

Assumption 5.1 (Critical Density Value)**.** *The critical density value is* $\rho_c \in (0, \frac{\rho_{\text{jam}}}{2})$.

Assumption 5.2 (Weight Balanced)**.** *The graph* G_{C} *is weight balanced including boundary conditions, or equivalently,* $K^\top \mathbf{1}_n + \alpha = \mathbf{1}_n$ *and* $K\mathbf{1}_n + \beta = \mathbf{1}_n$.

Assumption 5.3 (Existence of Sources and Sinks)**.** *There exists a source and sink with nonzero strength somewhere in* G_{C}.

Note that while these are strong assumptions, the following analysis helps clarify the overall behavior of the system dynamics. Each assumption is required in the proof of Theorem 5.4. Assumption 5.1 bounds the critical density value to be between reasonable values. Assumption 5.2 requires that sinks and sources have specific coefficients to enable free flow if possible. Assumption 5.3 allows for flow into and out of the system. Under these conditions, the following result holds.

Theorem 5.4 (Stability to critical density values)**.** *Given Assumptions 5.1, 5.2, 5.3, the dynamics of Equations* (3.3) *through* (3.6) *and a connected graph* G_{C}*, any initial state with* $\rho_i(0) \in [0, \rho_{\text{jam}}]$, $\forall\, i \in C$*, converges practically to* ρ_c*. In other words,* $\lim_{k\to\infty} \rho(k) \in [(\rho_c - \epsilon)\mathbf{1}_n, (\rho_c + \epsilon)\mathbf{1}_n]$*, where* $\epsilon \leq \frac{\Delta t v_f \rho_c}{L}$. □

Intuitively, this result holds due to the natural dynamics as well as the existence of ideal boundary conditions.

Proof. First, we check that $\rho^* = \rho_c \mathbf{1}_n$ is indeed an equilibrium. Combining Equations (3.3) to (3.6) gives

$$\rho_i(k+1) = \rho_i(k) + \frac{\Delta t}{L\ell}\left(\sum_{a \in \mathcal{N}_i^{\text{in}}} k_{ai} \min\{\mathbf{q}_a(k), \mathbf{q}_i(k)\} - \sum_{b \in \mathcal{N}_i^{\text{out}}} k_{ib} \min\{\mathbf{q}_i(k), \mathbf{q}_b(k)\} \right).$$

The first minimum term simplifies to $v_f \ell \rho_c$ if a is not a source, and $\alpha_a v_f \ell \rho_c$ if a is a source. Similarly, the second minimum term simplifies to $v_f \ell \rho_c$ if b is not a sink, and $\beta_b v_f \ell \rho_c$ if b is a sink. Setting $\rho_i(t+1) = \rho_i(t)$ to define an equilibrium results in $K^\top \mathbf{1}_n + \alpha = K\mathbf{1}_n + \beta$. This holds given Assumption 5.2, so $\rho = \rho_c \mathbf{1}_n$ is an equilibrium.

To prove convergence to this equilibrium, we define a Lyapunov fuction

$$V(\rho) = \min_{i \in C} \begin{cases} v_f(\rho_i - \rho_c), & \rho_i \le \rho_c, \\ \frac{v_f \rho_c}{\rho_{\text{jam}} - \rho_c}(\rho_c - \rho_i), & \rho_i > \rho_c. \end{cases}$$

This function is inversely proportional to the minimum flow rate in the system, is minimized at $\rho = \rho_c$, and is non-increasing along the dynamics assuming a small Δt, we will see this by bounding the flow in and flow out of each cell. For some state ρ, define $\underline{d} = V(\rho)/v_f$ and $\overline{d} = V(\rho)(\rho_{\text{jam}} - \rho_c)/(v_f \rho_c)$. Assume for now that $\rho_c \le \rho_i \le \rho_c + \overline{d}$ for some cell i. Then based on Equations (3.3) to (3.6),

$$\begin{aligned} v_f \ell(\rho_c - \underline{d}) &\le q_i^{\text{in}}(k) \le \frac{v_f \rho_c}{\rho_{\text{jam}} - \rho_c} \ell(\rho_{\text{jam}} - \rho_i(k)), \\ v_f \ell(\overline{d} - \rho_c) &\le q_i^{\text{out}}(k) \le v_f \ell \rho_c. \end{aligned}$$

This holds regardless of the number of in-neighbors, out-neighbors, sources, and sinks that are connected to i because of Assumption 5.2. Using these inequalities, we can

bound the density update as follows:

$$\rho_i(k+1) \leq \rho_i(k) + \frac{v_f \Delta t}{L}\left(\frac{\rho_c}{\rho_{\text{jam}} - \rho_c}(\rho_{\text{jam}} - \rho_i(k)) - (\rho_c - \overline{d})\right).$$

Under Assumption 5.1, $\frac{\rho_c}{\rho_{\text{jam}} - \rho_c} \leq 1$ which means $\rho_i(k+1) \leq \rho_c + \overline{d}$ if

$$\Delta t \leq \frac{L}{v_f}. \tag{3.8}$$

Similar arguments show $\rho_i(t+1) \geq \rho_c - \underline{d}$ when Equation (3.8) holds, so $V(\rho)$ is non-increasing. If initially $\rho_c - \underline{d} \leq \rho_i(t) \leq \rho_c$, analogous arguments using differing bounds on q_i^{in} and q_i^{out} lead to the same conclusion of Equation (3.8).

To prove a guaranteed decrease in $V(\rho(k))$, we must look the cardinality of $\Xi(k) = \{i \in C \mid \rho_i(k) = \max_{i \in C} \rho_i(k)\}$ and $\Phi(k) = \{i \in C \mid \rho_i(k) = \min_{i \in C} \rho_i(k)\}$, and prove that each cardinality will decrease.

Suppose there exists some $i \in \Xi(t) \cap \Xi(k+1)$ s.t. $\rho_i(k+1) = \rho_i(k) = \rho_c + \overline{d}$ with one out-neighbor j. This only occurs when $q_i^{\text{in}}(k) = q_i^{\text{out}}(k)$, and applying Equations (3.3) to (3.6), this requires $\rho_j(k) = \rho_c + \overline{d}$ where j is the out-neighbor of i. If j satisfies $\rho_j(k+1) = \rho_j(k) = \rho_c + d$ then we reapply this argument until we can find the front of group (defined when an out neighbor is not $\rho_c + \overline{d}$, or is a sink). For this cell h, we know $q_h^{\text{in}}(k) > q_h^{\text{out}}(k)$, so $\rho_h(k+1) < \rho_h(k)$ and Ξ loses a member at time $t+1$. It is possible for a node i s.t. $\rho_i(k) < \rho_c + \overline{d}$ to satisfy $\rho_i(k+1) = \rho_c + \overline{d}$, but this can only occur if each out-neighbor of i is at $\rho_c + \overline{d}$, this can be seen from Equation (3.4). So for each cluster of nodes in $\Xi(k)$, one (or more, at an intersection) node can join $\Xi(k+1)$, but each node that joins requires all of its out-neighbors to be at $\rho_c + \overline{d}$. On a single road side, one cell can join $\Xi(k+1)$ and one cell must leave $\Xi(k+1)$, at a normal intersection four cells can join $\Xi(k+1)$ and four cells must leave $\Xi(k+1)$, this can be also derived from Equations (3.3) to (3.6). This is a traffic congestion wave. Under Assumption 5.3, the

wave will eventually propagate to a source, and once the wave is adjacent to a source, the it must be $|\Xi(k+1)| < |\Xi(k)|$ because any cell leading into the wave will not achieve $\rho_c + d$. Analogous arguments hold for Φ. Once both $\Xi(k+1)$ and $\Phi(k+1)$ are empty, then $V(\rho(k+1)) < V(\rho(k))$. This holds for sufficiently large $V(\rho(k))$, so in summary $V(\rho(k))$ approaches a neighborhood around $v_f \rho_c$ asymptotically. □

We can make a similar Lyapunov argument proving that

$$\lim_{t\to\infty} \rho_i(t) \in [\rho_{\text{src}}^{\min}, \rho_{\text{snk}}^{\max}], \quad \forall\, i \in C,$$

where $\rho_{\text{src}}^{\min} = \rho_c \min_{i\in C}(\alpha_i + \sum_{j\in C} k_{ji})$ and $\rho_{\text{snk}}^{\max} = \rho_{\text{jam}} - \rho_c \min_{i\in C}(\beta_i + \sum_{j\in C} k_{ij})$. Bounding the flow rate in and out excludes any state from being at equilibrium outside these bounds, given the previous assumptions.

Remark 5.5. *Given additive zero-mean i.i.d. noise on the dynamics equation* (3.6)*, it is easy to see that* $V(\mathbb{E}[\rho(k+1)]) \leq V(\rho(k))$*. The effect of this noise on convergence to equilibrium is tested in simulation, see Figure 3.8.* □

Remark 5.6. *The previous analysis helps motivate the rerouting algorithm. Maintaining the sufficient condition Assumption 5.2 is crucial for equilibrium behavior of the network, and it is expected that, when boundary conditions oscillate about this condition, convergence to a close-to-equilibrium condition will occur. However, within these constraints we would like to hasten convergence to further reduce delays, the benefits of which are heightened under time-varying boundary conditions.*

3.5.2 Rerouting Algorithm

In some situations, it is feasible to direct vehicles where to go, in order to maintain a balanced network over time. One example is a near future road setting where driverless

vehicles can be dynamically reassigned to pick up waiting passengers. Freight-type of vehicles or autonomous cars in mobility-on-demand systems could also be influenced in real traffic based on AIM-vehicle communication mechanisms. Control over vehicle direction means that K can be altered under some constraints to improve the total flow over the time interval. Because maximum flow is achieved when $\rho_i = \rho_c$, $\forall\, i \in C$, flow through intersections can be redirected from more dense roads to less dense roads, keeping the system close to ρ_c. This problem is formulated in Problem 2.

A greedy approach suits this problem because it increases immediate flow through the intersection and speeds up the balancing of neighboring roads while maintaining the equilibrium of the natural dynamics. A potential strategy is to redirect flow from all in-neighbors of an intersection to the least dense out-neighbor. If the least dense out-neighbor is more dense than ρ_c, then redirect towards a sink if possible. However, this approach creates congestion when sinks are not ideal by attempting to direct flow out of the system but the sink restricting the flow out, so we require an alternative policy. Instead, each out-neighbor is sorted according to how much flow they will allow in, and they are paired with in-neighbors which provide the most flow, see Rerouting Algorithm.

This algorithm is decentralized, the only information required is from immediate neighbors of an IM. Intuitively, the algorithm is improving flow by directing flow from the most dense in-neighbors of the intersection to the least dense out-neighbors so that the flow from/to the most/least congested road is relatively unrestricted. This pushes both states more quickly towards ρ_c, and because they are the furthest away from ρ_c, this helps decrease $V(\rho(k))$ more rapidly. Under varying boundary conditions, this improvement is enhanced.

Algorithm 6: Rerouting Algorithm

for *each intersection* $z \in Z$ *at each time* k **do**
  $D^+ \leftarrow$ vector of cells flowing into z, sorted by decreasing density;
  $D^- \leftarrow$ vector of cells receiving flow from z, sorted by increasing density;
  for $\gamma \in \{1, \ldots, |D^+|\}$ **do**
    $x \leftarrow D^+_\gamma$;
    $y \leftarrow D^-_\gamma$;
    for $\zeta \in \{1, \ldots, n\}$ **do**
      if $k_{x,\zeta} > 0$ *and* $\zeta \neq y$ **then**
        $k_{x,\zeta} \leftarrow k_{\min}$;
      end
      if $k_{\zeta,y} > 0$ *and* $\zeta \neq x$ **then**
        $k_{\zeta,y} \leftarrow k_{\min}$;
      end
    end
    $k_{x,y} \leftarrow 1 - (|D^+| - 1)k_{\min}$;
  end
end

3.6 Simulation Results

In simulation, both lane reversal and rerouting vehicles reduce overall traffic delay under imbalanced conditions as we discuss next. We use $\Delta t = 1$ second, $L = 500$ meters, $\ell = 4$ lanes per road, and $v_f = 60$ km/hr, which also satisfy Equation (3.8). Note, $\rho_{\text{jam}} \approx 226 \frac{\text{veh}}{\text{km·lane}}$ was calculated by assuming an average vehicle length of 4.11 meters and an average gap between stationary vehicles of 0.31 meters. We chose $\rho_c = \rho_{\text{jam}}/3$.

3.6.1 Lane Reversal Algorithm

Generally, lane reversal creates a significant short-term improvement of traffic flow. Lane reversal can hinder overall traffic flow in the long-term when reversing a lane requires lane merging somewhere else in the system or when the road side with less lanes receives heavy traffic flow afterwards. This first issue can be addressed through coupling roads together so that their control variables are equal and no lane merging is

required between them, in simulation we choose the more conservative control value from both intersections and apply that to both roads. The second issue can be at least partially addressed through predicting traffic patterns using past data or communications with a larger portion of the road network, though we do not address this in this chapter.

We simulated a two road network with an intersection between them, sources and sinks at the boundary, and an initial state which had light congestion randomly sampled from $[0,\rho_{\text{jam}}/2]$ on one side of both roads and heavy congestion randomly sampled from $[\rho_{\text{jam}}/2,\rho_{\text{jam}}]$ on the other. Instead of optimizing over a time horizon, a greedy algorithm was enough to see significant improvements. The boundary conditions α and β on one end of the network were randomly sampled from $[0, 1]$ and $\alpha = 1$ and $\beta = 1$ on the other side, creating an imbalanced flow. U-turns do not occur, and flow from each road to any neighbor is equally likely. One example of the benefit of lane reversal on the objective function is shown in Figure 3.6, it is evident that the flow rate increases immediately after lane reversal. In this figure, the solid line represents lane reversal and the dashed represents no lane reversal, note the immediate improvement. In this example, one lane was reallocated from westbound to eastbound at $t = 0$ then the state remained constant.The throughput improvement of 100 experiments is shown in Figure 3.7, we can see a large variance in the data but there is always significant improvement. Note, maintaining a constant number of lanes along all paths becomes impossible with a larger system, creating merges which can negatively affect the overall equilibrium.

We also added zero-mean Gaussian noise to Equation (3.6) to check the robustness of Theorem 5.4 (Stability to critical density values), the effect of noise on the two road network is seen in Figure 3.8. Deviation from equilibrium was averaged over each cell and over 120 time steps, with the initial state at equilibrium. With zero noise there is no deviation from equilibrium, and as the noise level increases, the average deviation from equilibrium increase quite linearly from small noise values.

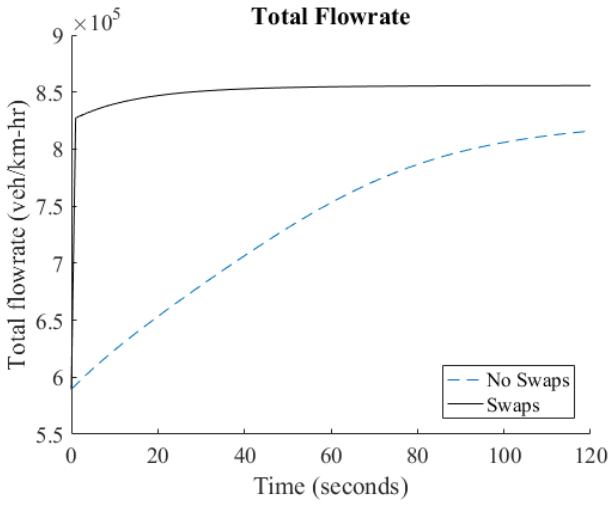


Figure 3.6: Flow rate using lane reversal on two roads

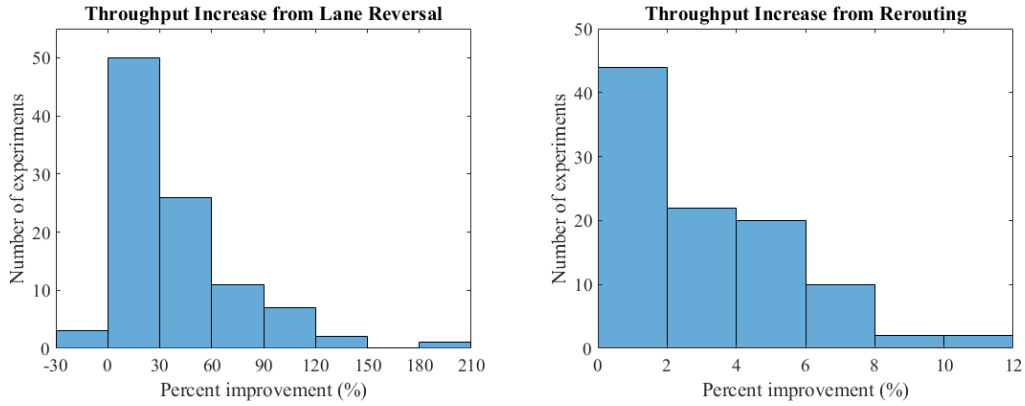


Figure 3.7: Lane reversal and rerouting performance

3.6.2 Rerouting Algorithm

The rerouting policy improves flow rate in both short term and long term while maintaining the original equilibrium, though in a less dramatic fashion than lane reversal. We have simulated a two block by two block road network with random initial densities under random constant boundary conditions with $k_{\min} = 0.05$. We implemented the rerouting policy on the central intersection, see Figure 3.9, black represents very dense and white represents no vehicles. This shows an initially unbalanced state which converged to a more efficient state with help from the rerouting policy in the middle

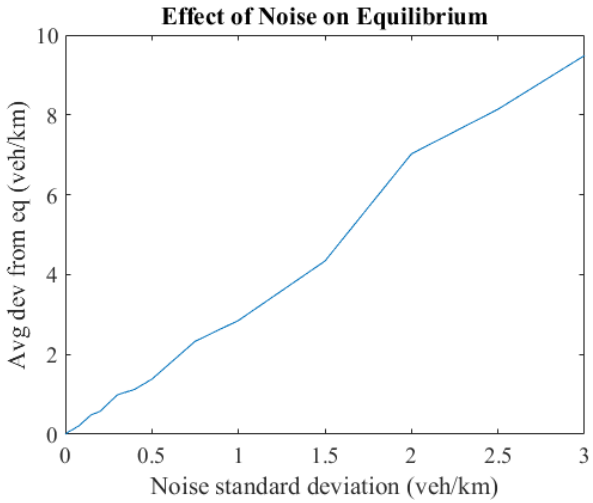
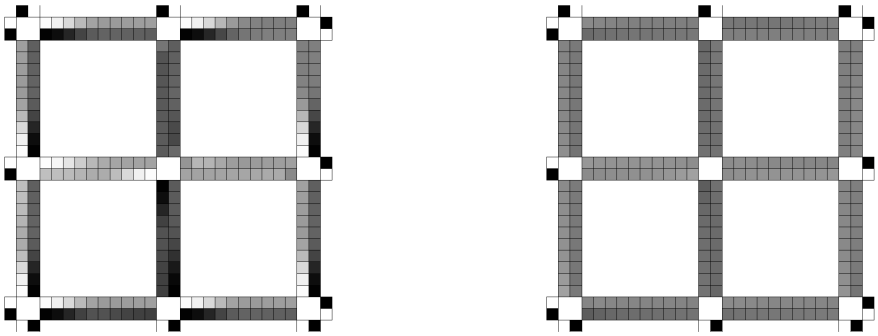


Figure 3.8: Effect of noise on equilibrium

intersection, see 3.10. The dashed line represents rerouting and the solid line represents no rerouting, note the consistent improvement.



$t = 0$ seconds $t = 20$ seconds

Figure 3.9: Road network density

In Figure 3.7, we can see relative improvements on the total flow rate given by this policy, ranging from negligible to moderate. We note that the objective function describes the performance of the each cell in the system, and in this network there are 240 cells, only 8 of which are connected to the intersection performing the policy, so only moderate improvement is expected. Under specific initial conditions, the most improvement was seen at 36%, on average improvement is between 0 and 8%. In every

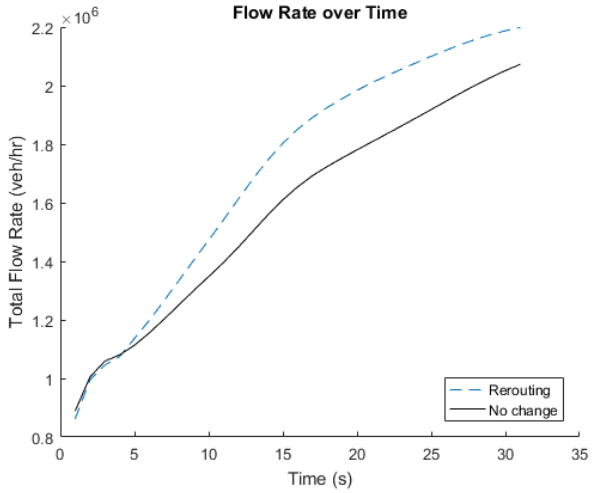


Figure 3.10: Flow rate using lane reversal on two roads

case, this algorithm improved overall flow rate.

3.7 Conclusion and Future Work

In conclusion, we extended the cell transmission model and established objective functions with the goal of minimizing total time spent on the road. We proposed a distributed algorithm to efficiently calculate and implement an appropriate lane direction reallocation. We also proposed a distributed algorithm to dynamically reroute vehicles to improve the long term behavior of the system. We proved convergence of the lane reversal algorithm to a critical point and bound the equilibria of the traffic rerouting algorithm under certain conditions. We showed through simulations performance gains using lane reversal on a network under particular conditions and using rerouting under different conditions.

There are many avenues for future work on this problem. Another approach based on the idea of synchronizing a system of intersections to some "mean intersection" behavior in order to achieve adequate coupling [37] could be attempted. One avenue is improving the traffic model and comparing its predictions with real traffic traffic data

to ensure accuracy, in particular the vehicle time spent in an intersection is currently assumed constant. A microscopic model will better capture the dynamics of real vehicles on a road network, for example by implementing simulations which include spawning and tracking individual vehicles. Measurement and model uncertainty could be characterized for more accurate estimation. Reinforcement learning can address some of the issues with using a greedy lane reversal algorithm to further reduce total time delays.

Chapter 3, in part, has been published in proceedings of Mathematical Theory of Networks and Systems, Minneapolis, MN, USA, July 2016, by E. Gravelle and S. Martínez, and is submitted for publication to International Journal of Control, 2017, by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Using time-inconsistent wait-time functions for cycle-free coordinated traffic intersections

4.1 Abstract

Sensing and computing capabilities of modern traffic intersections have greatly improved in recent years, but current control policies do not fully utilize these capabilities. In this chapter, we present a novel intersection control algorithm based on an objective function that accounts for drivers' time preferences. In particular, the intersection places greater importance on a vehicle which has been waiting at an intersection over one that just arrived. Coordination between intersections is achieved through added a term to the objective function using the green wave idea. Under this policy and a macroscopic dynamical model, we provide a sufficient condition for the controller to maintain uniformly bounded weighted queues at an intersection given sufficiently small spawn rates. We test our algorithm and results extensively in a realistic microscopic

simulation, through measuring queue stability and various performance metrics.

4.2 Introduction

Modern advancement of technology combined with the rapid decrease in cost of processors has exponentially increased the capabilities of common infrastructure systems over recent decades. One such system is the traffic intersection manager, which controls the phases of a traffic light, allowing vehicles in certain lanes to pass safely through the intersection. The estimation of traffic around intersections together with the use of adaptive control algorithms can lead to significant improvements on traffic conditions and quality of service. Most modern intersections use loop detectors in the road to approximate the position, velocity, and/or density of vehicles and make control decisions. However, loop detectors are expensive to install and maintain, a significant fraction of loop detectors are not functioning on a given day [39], [38], and they only provide data at discrete points on the road. Improved camera quality as well as tracking algorithms has opened the door for traffic intersections to stop relying on loop detectors and start using cameras, offering a more robust and rich source of information. Motivated by this scenario, we design a novel intersection control policy utilizing the position and velocity information of each vehicle to enable a more intelligent policy.

One common algorithm for intersection control is SCOOT, or Split Cycle Offset Optimization Technique [24]. This controller adjusts the split, cycle, and offset parameters of the cyclical sequence of phases to accomodate instantaneous traffic demands as read by loop detectors. However, there are a few problems with this formulation which many alternatives also fail to address [31], [27]. One issue is that the parameters which SCOOT optimizes perturb the cycle in a limited fashion, so SCOOT is quite constrained and efficiency is lost. Another limitation is given by the assumption that

each vehicle is just as important as the next, which is implicit in the objective. We claim that a better objective accounts for the time that a vehicle has been waiting so that the relative importance of one vehicle which has been waiting is more than one vehicle which has just arrived. Intuitively this idea makes sense as a mechanism to ensure that every vehicle eventually crosses the intersection, studies have also shown that humans value time inconsistently [28] and that uncertainty in travel time is costly [34], both of these issues are mitigated by this new objective.

Some more recent works have extended these classical algorithms. Backpressure routing techniques are applied to traffic systems using fixed time slots without considering phase transition times in [46]. Linear-quadratic regulator theory is applied in [13] with an emphasis on heavily congested conditions, which does not consider the effect of offset of consecutive junctions. A multi-objective linear programming approach which considers light switchover dates in a time horizon is described in [16]. A genetic algorithm is applied to maximize traffic flow using a graphics processing unit in [41]. Work has also been done on bandwidth maximization using variable speed limits [35] and partitioning techniques [43]. However, none of these works fully utilize modern sensing technology in a flexible optimization setting, which can be used to e.g. prioritize those who have been waiting longer times.

4.2.1 Contributions

In this chapter, we devise a traffic intersection policy to minimize an objective function that accounts for time-inconsistent waiting-time functions. These waiting functions have similar properties to hyperbolic discount functions, providing a simple model for human rewards, and seem to be effective to control traffic at intersections. In particular, the objective considers incoming vehicles from neighboring intersections, and places additional weight on a vehicle which has been waiting some time, this is

representative of human preferences. We consider a class of policies that can be effective for controlling traffic at intersections and keep traffic bounded at intersections. We provide a sufficient condition on the intersection switching rule using a macroscopic flow model that ensures uniform bounded weighted queue lengths under a small enough vehicle spawn rate. We also add a term to the objective function to enable cooperation between intersections with the goal of increasing overall efficiency. In simulation, we model vehicle agents after real drivers: a driver attempts to drive its route at the speed limit while following traffic laws and maintaining adequate distance between it and other vehicles. We simulate our intersection controller policy with the more complex vehicle agents on a system with two intersections and show the benefits of the new control algorithm. In particular, we verify the bounded weighted queue result for the complex model, and we show how the policy effectively minimizes the total squared waiting time or human impatience at the intersection. Simulation results verify how the acyclic policy is more effective in showing bounded weighted queues than a cyclic policy counterpart, and show the effect of the coordination term on various performance indices.

4.2.2 Outline

This chapter is organized as follows. Section 4.3 contains the problem formulation with motivation and the vehicle behavior model. Section 4.4 contains the queue stability theorem and proof under a simplified macroscopic model. Section 4.5 contains simulation details and numerical and plotted results. Finally, Section 4.6 contains a summary of our work and a discussion of future work.

4.3 Problem Formulation

In this section, we formulate the intersection control problem by defining the objective function and control variables. The goal of the intersection policy is to minimize an objective function which characterizes travel time of all vehicles while ensuring that vehicles rarely have wait for a long time at a given intersection.

4.3.1 Single intersection problem

An intersection I is a junction of road segments of given length $L_I > 0$. These roads can have any number of lanes and/or feasible allocation of turn lanes. We denote by N the total number of vehicles that flow into the intersection system over a time horizon, $[0, H]$, which is discretized as $h = k\Delta h \leq H$, for $k \in \mathbb{N}_0$ where $\mathbb{N}_0$ is the set of natural numbers including zero and $\Delta h > 0$ as a small discretization step.

Let $i \in \{1, \ldots, N\}$ denote a vehicle passing through the intersection over this time horizon. As a first approximation to the intersection problem, consider the cost function:

$$\underset{\mathbf{u} \in \mathcal{U}^{H\Delta h}}{\text{minimize}} \quad \sum_{i=1}^{N} (\Delta t_i + w(\hat{t}_i)).$$

The variable Δt_i (with $\Delta t_i \approx k_i \Delta h$, for some k_i) is the time that vehicle i would spend at the junction if it were to move through the system without stopping, $\hat{t}_i$ (with $\hat{t}_i \approx \hat{k}_i \Delta h$, for some $\hat{k}_i$) is extra time spent by vehicle i due to the delay at I, and $w : [0, +\infty) \to [0, +\infty)$ is a class $\mathcal{K}_\infty$ weighting function which penalizes long delays. A function $f : [0, \infty) \to [0, \infty)$ is said to belong to class $\mathcal{K}_\infty$ if it is strictly increasing, $f(0) = 0$, and $\lim_{r \to \infty} f(r) = \infty$. The set $\mathcal{U}$ is the set of phases by which a conflict-free set of lanes is selected at a given time. For example, one possible phase would allow all vehicles travelling straight through the intersection north or south to pass through the intersection safely, while all other vehicles would be prohibited. The sequence $\mathcal{U}^{H\Delta h}$ is a sequence of phases

for $H\Delta h$ time steps. For ease of notation and without loss of generality, we consider $\mathcal{U} \triangleq \{N_l S_l, N_l N_s, N_s S_l, S_l S_s, W_l E_l, W_l W_s, W_s E_l, E_l E_s\}$, where the base variable represents whether the road originates from north, south, east, or west, and the subscript denotes left-turn lanes (l) or straight/right lanes (s). The extra time $\hat{t}_i$ is incremented when $v_i \leq \gamma v_f$, where v_i is the velocity of vehicle i, $\gamma \in [0, 1]$ is the wait threshold which defines when a vehicle is waiting, and v_f is the speed limit.

We assume that the intersection can track vehicles within some distance L which are in lanes moving towards it, so it knows time-of-sighting (measured in its time clock t_{sys}), position, and velocity of each vehicle. Estimating $\hat{t}_i$ for all vehicles is difficult because the dynamics of each vehicle must be calculated forward in time until they leave the system, and this is both error-prone due to many influences on the dynamics and computationally expensive. Thus, we simplify the original problem above by proposing a simpler objective that does not require explicit travel time estimates. More precisely, we consider only the current time step, and focus on the current cost of delays of each lane that are weighted to determine which directions to allow through. In other words, we consider:

$$\underset{u \in \mathcal{U}}{\text{minimize}} \quad \sum_{\ell \in u} w_\ell(\hat{t}_i) \triangleq \sum_{\ell \in u} \sum_{i \in \ell} w(\hat{t}_i), \tag{4.1}$$

where w_ℓ is the sum of weights of vehicles in lane ℓ which are allowed through in phase u. This problem is solved at every time instant $h \geq 0$, and a switch from a u_0 to a u_1 is implemented provided the following additional constraints hold.

First, a substantial difference in lane weights is required to be able to switch to a new phase due to delays associated with changing phases including yellow lights and vehicle acceleration time. So a change is triggered from u_0 to u if

$$\sum_{\ell \in u} w_\ell(\hat{t}_i) > \eta \sum_{\ell \in u_0} w_\ell(\hat{t}_i), \tag{4.2}$$

holds, for some $\eta > 1$, where η is a tuning parameter called the switch threshold. In other words, this is a constraint to our Problem (4.1).

In addition, we require a lower bound on the time spent at a specific state u_0 to consider the time it takes for vehicles to accelerate from rest. We define this bound based on getting a certain number of vehicles through the intersection from rest. We define n_ℓ^I as the minimum number of desired vehicles to let through at each lane ℓ at intersection I. Then, $n_u^I = \max_{\ell \in u} n_\ell^I$ is the minimum number of desired vehicles to allow through during control u, based on which lanes will be allowed through by u. With knowledge of the queue lengths at each lane Q_ℓ^I, define $Q_u^I = \max_{\ell \in u} Q_\ell^I$ and the minimum time in a phase is given by

$$g_u^I = (\min\{Q_u^I, n_u^I\} - 1)t_{\text{gap}} + \sqrt{\frac{2L_u^I}{a}}, \tag{4.3}$$

where t_{gap} is the time between vehicles moving safely at maximum velocity, L_u^I is the maximum length of the path through the intersection during phase u, and a is the average acceleration of vehicles. Let kh be a discretized time instant, for some $k \in \mathbb{N}$. Suppose that $k_0 h$ was the last time a phase change was implemented. Then, the optimization problem (4.1) need not be re-solved unless $(k - k_0)\Delta h \geq g_{u_0}^I$.

4.3.2 Multiple intersection problem

Given a system of multiple intersections, a naive extension would partition the roads between the intersections and assign each part to the nearest intersection, serving to decouple the intersections and localize their control. To avoid a complete decoupling, which could have detrimental effects on joint performance, we extend the one-intersection problem formulation by adding a term to enable cooperation using the idea of *green wave* [35]. A green wave results from the offset synchronization of green traffic lights along a road, which allows vehicles within a certain bandwidth to pass through multiple

intersections without stopping, we call this group of vehicles a platoon. This requires some communication between the intersections, however at most only a few floating point numbers are required to be transmitted every second. Current methods which attempt to maximize this bandwidth usually require a fixed cycle length. To incorporate this idea in a more flexible non-cyclic framework, we introduce a coupling term. The problem now is extended to

$$\begin{aligned} \underset{u\in\mathcal{U}}{\text{minimize}} \quad & \sum_{\ell\in u}\left(w_\ell(\hat{t}_i)+B_\ell\right), \\ \text{subject to} \quad & u\in\mathcal{P}, \end{aligned}$$

where B_ℓ is a term that exploits the knowledge of future vehicles either from intersections or just from additional sensors to plan ahead, and $\mathcal{P}$ is the set of phases which satisfies the constraints in Equations (4.2) and (4.3). We assume that intersections are synchronized, and that an intersection I will have some information about when vehicles will be arriving in the future from another intersection. We assume it knows $\{\ell_p, E_p, T_p\}$, where p denotes a platoon and ℓ_p, E_p, T_p are the lane, the number of vehicles coming, and the expected time of arrival, respectively. A transition time between each phase is required, which is simply the time a light must stay in yellow before transitioning, we denote this as Y_p^I. We define ξ_p^I as the amount of time required for all vehicles in platoon p to cross the intersection I, and we let $z_p^I = t_{\text{sys}} + Y_p^I + S - T_p^I$ represent a centered time, where S is the average vehicle's time to stop from full speed and t_{sys} is the system time shared by all intersections. In this way, $z_p^I = 0$ means that the platoon arrives and sees a green light simultaneously.

With knowledge of this data, a benefit is achieved if the light is green when the first vehicle arrives, and stays green for the whole platoon to make it through the

intersection. With this, define

$$B_\ell = \alpha E_\ell \max\left\{0, \min\left\{\frac{z_p^I}{\xi_p^I} + 1, \min\left\{1, \frac{g_u^I}{\xi_p^I}\right\}, \frac{g_u^I - z_p^I}{\xi_p^I}\right\}\right\}, \tag{4.4}$$

where $\ell = \ell_p$, α is a control parameter determining how heavily to consider the platoons, see Figure 4.1. This term is maximized when the platoon overlaps completely with a green light and is zero when there is no overlap. Inside the minimum, $\frac{z_p^I}{\xi_p^I} + 1$ is the fraction of the platoon which benefits from a switch to green when the platoon arrives early, $\min\{1, \frac{g_u^I}{\xi_p^I}\}$ characterizes the maximum fraction of the platoon which can cross during the green, and $\frac{g_u^I - z_p^I}{\xi_p^I}$ is the fraction of the platoon which benefits from a switch to green when the platoon arrives late. Intuitively, B_ℓ is minimized and therefore provides the most benefit when the green phase coincides with the arrival of vehicles, and is maximized and provides no benefit when the green phase has no overlap with vehicle arrival.

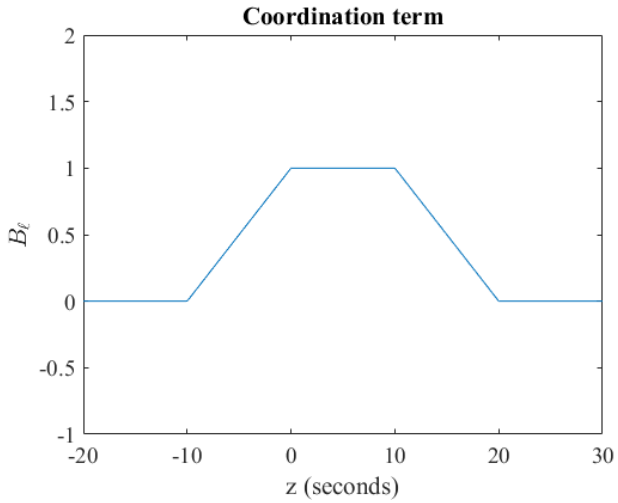


Figure 4.1: Plot of coordination term

4.3.3 Vehicle Behavior Model

In our subsequent simulations, we modeled vehicle agents' behavior after real drivers, according to the following rules:

(i) A vehicle has a fixed path through the system, from origin at a boundary to destination at a boundary.

(ii) Lane changes are not allowed.

(iii) A vehicle accelerates uniformly up to the speed limit unless:

- there is risk of collision with another vehicle, with an added safety buffer,
- there is a red or yellow light and the vehicle can safely stop in front of an intersection,

in which the vehicle decelerates uniformly.

4.3.4 Wait-Time Function

The weight functions model the wait-time of drivers at the intersection. The functions we consider here are related to hyperbolic, time-inconsistent discount functions in reward maximization problems [28]. Unlike exponential discouting, these functions reflect the human preference to choose smaller-but-sooner rewards over larger-but-later rewards as the delay occurs sooner rather than later in time. Hyperbolic discount functions in reward maximization problems take the form $f(\hat{t}) = \frac{1}{\alpha+\phi\hat{t}}$, where α and ϕ are some fixed coefficients, and $\hat{t}$ is the delay. In our cost-minimization setting, this translates into wait functions of the form $w(\hat{t}) = \alpha + \phi\hat{t}$. With a similar property to these, we consider functions of the form

$$w(t) = \phi t^2 + \alpha t + \beta, \tag{4.5}$$

and, for simplicity in the computations and simulations that follow, $w(t) = \phi t^2$ for some constant $\phi > 0$. These functions preserve the hyperbolic discounting property, while performing better than purely hyperbolic functions in maintaining bounded weighted queues in traffic simulations. Additionally, the functions ensure that no vehicle is stuck indefinitely, and should help reduce the average travel time variance compared to a linear model.

4.4 Queue Stability

In this section, we prove weighted queue length stability under a simplified model. Weighted queue length for road r is defined as w_r. System stability is achieved when all lane queue lengths remain bounded below some value such that each queue does not spillover into other intersections.

The following assumptions were made to construct the new model:

Assumption 4.1 (Traffic model)**.** *Traffic flow follows a continuous-time macroscopic model where change in vehicle density on some part of the road is proportional to the net flow rate in, or* $\rho(t+\Delta t) = \rho(t) + \Delta t(q^{in} - q^{out})$*. Density near an intersection is proportional to the queue length. There is a constant flow rate* $q_{in,r} = q^{in}$ *into each non-outbound road* $r \in \{1, \ldots, 4\}$ *from the boundary. Given a queue of vehicles with the freedom to move, the flow rate at the front of the queue is* $q^{out} > q^{in}$*.*

Assumption 4.2 (Two-phase intersections)**.** *There are two intersection phases, each allowing bi-directional flow on the two roads, one running north-south and the other east-west. Each phase has minimum duration of* $g_{\mathrm{NS}}, g_{\mathrm{EW}}$*, respectively. Flow begins immediately when a new phase begins.*

Assumption 4.3 (Linear wait time)**.** *The average time that a vehicle has been waiting at*

an intersection is proportional to the number of vehicles in the queue, n_r. This means $w_r(t) \propto n_r^2$ for road r.

We formed a simpler dynamical model and applied Algorithm 7 in order to theoretically ensure stability. The following theorem contains the stability result:

Theorem 4.4. *Given Assumption 4.1 (Traffic model) with constant flow-rate parameters $q_{in,r}$, $\forall r \in \{1,\dots,4\}$ and q^{out}, Assumption 4.3 (Two-phase intersections), Assumption 4.3 (Linear wait time), and sufficiently small spawn rate $q_{in,r} = q^{in}$, $\forall r \in \{1,\dots,4\}$, any finite switch threshold $\eta > 1$ will ensure that the intersection control solution of Equation* (4.1) *under constraints* (4.2) *and* (4.3) *maintains uniformly bounded weighted queues in all lanes for all time.*

Interpreting this result leads to the following intuitive explanation: η is directly related to T, and as long as η is small enough so that that the growing weighted queues remain bounded and large enough so that the minimum green time does not affect the dynamics, the system is stable.

Proof. Road indices 1, 2, 3, 4 indicate north, east, south, west, respectively, relative to the center of the intersection. For now assume initial conditions such that the north-south road has a green light $u(0) = [1,3]$, n_r is the number of vehicles on road r, $w_{\mathrm{NS}}(0) = \eta w_{\mathrm{EW}}(0)$, and when a switch occurs at time T, $T \geq g_{\mathrm{NS}}$. Given any initial phase $u(0)$, a cycle with duration $t_{\mathrm{cycle}} > 0$ occurs when $u(t_{\mathrm{cycle}}) = u(0)$ with $u(\bar{t}) \neq u(0)$ for some $\bar{t} \in (0,T)$. Queue lengths will remain bounded if, $n_r(t + t_{\mathrm{cycle}}) \leq n_r(t)$, $\forall r \in \{1,\dots,4\}$. Define $w_{\mathrm{EW}}(t) = w_1(t) + w_3(t)$ and $w_{\mathrm{NS}}(t) = w_2(t) + w_4(t)$. Then due to Assumption 4.3 (Linear wait time),

$$w_{\mathrm{EW}}(t) = \frac{n_2(t)^2}{2q_{\mathrm{in},2}} + \frac{n_4(t)^2}{2q_{\mathrm{in},4}} = \frac{(n_2(0) + q_{\mathrm{in},2}t)^2}{2q_{\mathrm{in},2}} + \frac{(n_4(0) + q_{\mathrm{in},4}t)^2}{2q_{\mathrm{in},4}},$$

where

$$n_r(0) = \sqrt{2q_{\text{in},r}w_r(0)},\ \forall\, r \in \{1,\ldots,4\}.$$

Simplifying,

$$w_{\text{EW}}(t) = w_2(0) + w_4(0) + \left(\sqrt{2q_{\text{in},2}w_2(0)} + \sqrt{2q_{\text{in},4}w_4(0)}\right)t + \frac{q_{\text{in},2}+q_{\text{in},4}}{2}t^2.$$

Now define $\Delta q_r = q_{\text{in},r} - q_{\text{out,r}},\ \forall\, r \in \{1,\ldots,4\}$. Computing $w_{\text{NS}}(t)$,

$$w_{\text{NS}}(t) = \frac{n_1(t)^2}{2q_{\text{in},1}} + \frac{n_3(t)^2}{2q_{\text{in},3}} = \frac{(n_1(0)+\Delta q_1 t)^2}{2q_{\text{in},1}} + \frac{(n_3(0)+\Delta q_3 t)^2}{2q_{\text{in},3}}.$$

Simplifying,

$$w_{\text{NS}}(t) = w_1(0) + w_3(0) + \left(\Delta q_1 \frac{\sqrt{2w_1(0)}}{q_{\text{in},1}} + \Delta q_3 \frac{\sqrt{2w_3(0)}}{q_{\text{in},3}}\right)t + \left(\frac{\Delta q_1^2}{2q_{\text{in},1}} + \frac{\Delta q_3^2}{2q_{\text{in},3}}\right)t^2.$$

One condition guaranteeing the previous definition of bounded weighted queues is $w_{\text{EW}}(\tilde{T}) = w_{\text{EW}}(0)$ and $w_{\text{NS}}(\tilde{T}) \le w_{\text{NS}}(0)$ at some time $\tilde{T}$. Due to the symmetry of the two phases and the immediate flow condition after a switch of Assumption 4.2, an equivalent condition is $w_{\text{EW}}(T) = w_{\text{NS}}(0)$ and $w_{\text{NS}}(T) \le w_{\text{EW}}(0)$ for some time T. Assume the phase changes when $w_{\text{EW}}(T) = \eta w_{\text{NS}}(T)$, so after applying these equations, we get

$$0 = w_2(0) + w_4(0) - \eta w_{\text{NS}}(T) + \left(\sqrt{2q_{\text{in},2}w_2(0)} + \sqrt{2q_{\text{in},4}w_4(0)}\right)T + \frac{q_{\text{in},2}+q_{\text{in},4}}{2}T^2, \tag{4.6}$$

$$0 \ge w_1(0) + w_3(0) - w_{\text{EW}}(0) + \left(\Delta q_1 \frac{\sqrt{2w_1(0)}}{q_{\text{in},1}} + \Delta q_3 \frac{\sqrt{2w_3(0)}}{q_{\text{in},3}}\right)T + \left(\frac{\Delta q_1^2}{2q_{\text{in},1}} + \frac{\Delta q_3^2}{2q_{\text{in},3}}\right)T^2. \tag{4.7}$$

Plugging in initial conditions to Equation (4.6) gives

$$0 = (1-\eta^2)(w_2(0)+w_4(0)) + \left(\sqrt{2q_{\text{in},2}w_2(0)} + \sqrt{2q_{\text{in},4}w_4(0)}\right)T + \left(\frac{q_{\text{in},2}+q_{\text{in},4}}{2}\right)T^2.$$

Define

$$\begin{aligned} A_1 &= \left(\frac{q_{\text{in},2}+q_{\text{in},4}}{2}\right), \\ B_1 &= \left(\sqrt{2q_{\text{in},2}w_2(0)} + \sqrt{2q_{\text{in},4}w_4(0)}\right), \\ C_1 &= (1-\eta^2)(w_2(0)+w_4(0)), \end{aligned}$$

then

$$T = \frac{-B_1 + \sqrt{B_1^2 - 4A_1C_1}}{2A_1}, \tag{4.8}$$

and because $\eta > 1$, $A_1 > 0$, $B_1 > 0$, and $C_1 < 0$, a real positive solution exists for T. This quantity represents how much time passes before $w_{\text{EW}}(T) = w_{\text{NS}}(0)$.

Equation (4.7) is satisfied when

$$\begin{aligned} T &\leq \frac{-B_2 - \sqrt{B_2^2 - 4A_2C_2}}{2A_2}, \\ A_2 &= \left(\frac{\Delta q_1^2}{2q_{\text{in},1}} + \frac{\Delta q_3^2}{2q_{\text{in},3}}\right), \\ B_2 &= \left(\Delta q_1 \frac{\sqrt{2w_1(0)}}{q_{\text{in},1}} + \Delta q_3 \frac{\sqrt{2w_3(0)}}{q_{\text{in},3}}\right), \\ C_2 &= (\eta-1)(w_2(0)+w_4(0)). \end{aligned} \tag{4.9}$$

Due to the fact that $\Delta q_r < 0$, $\forall r \in \{1,\ldots,4\}$ from Assumption 4.1 (Traffic model), $A_2 > 0$, $B_2 < 0$, and $C_2 > 0$, the solution to Equation (4.9) is real and positive when

$$\begin{aligned} B_2^2 - 4A_2C_2 &\geq 0, \\ B_2 + \sqrt{B_2^2 - 4A_2C_2} &< 0, \end{aligned}$$

both of which are satisfied when $q_{\text{in},r}$ is small enough, for all $r \in \{1,\dots,4\}$.

This argument can be repeated for each future switch by simply replacing instances of w_{EW} with w_{NS} and vice versa, due to $q_{\text{in},r} = q^{\text{in}}, \forall r \in \{1,\dots,4\}$. Now relaxing the assumption on initial conditions, in finite time one of two scenarios can occur. One is, the light will switch due to the weights so either $w_{\text{NS}}(T) = \eta w_{\text{EW}}(T)$ or $w_{\text{EW}}(T) = \eta w_{\text{NS}}(T)$, this is treated as the new initial condition and the argument holds. The second scenario is that the light will switch at g_{NS} due to the green time constraint, and $w_{\text{EW}}(g_{\text{NS}}) = w_{\text{EW}}(T) + \delta$. However, from here either a switch will occur due to the weights or a switch will occur due to the constraint from Equation (4.3), either case will results in queues bounded by $w_{\text{EW}}(T) + \delta$, and this argument can be propogated forward in time as well.

□

Remark 4.5. *If Assumption 4.1 (Traffic model) is relaxed and* $q_{in,r}(t) \in [q_{min}, q^{max}], \forall\, r \in \{1,\dots,4\}, \forall\, t$, *then the solution to Equation* (4.8) *is bounded (due to convexity) by the solutions to* (4.8) *with* $q_{in,r} = q_{min}$ *and* $q_{in,r} = q^{max}, \forall\, r \in \{1,\dots,4\}$. *Similarly, if* q^{out} *must ramp up to the maximum value from 0, a similar argument can be made, leading to similar results and overall stability but with tighter bounds on* η.

If we relax Assumption 4.3 (Linear wait time), the same ideas and mathematical principles can be applied to the more complex case where there are four possible phases (left-left, left-straight, left-straight, straight-straight) instead of one (straight-straight) in each direction. In the multiple intersection case, each individual intersection can be viewed as discussed previously, but now with a perturbation in the form of the cooperation term B_ℓ. *This perturbation is bounded as seen in Equation* (4.4)*, and in the worst case where many lanes have long queues, the perturbation will be small compared to the quadratic wait-time costs, so we expect bounded weighted queues in these scenarios as well.*

4.5 Simulation Results

The algorithm applied in simulation is seen in Algorithm 7. An intersection knows which lanes are associated to which phases, and has parameters fixed. At each time step, the weights are calculated in each lane, and these weights are associated to different phases. If there exists another phase with η times the weight of the current phase, a transition occurs to this phase. If there exists another phase which shares a direction with the current phase and has $\eta/2$ times the weight of the current phase, a partial transition occurs to this phase.

Algorithm 7: Intersection Control Algorithm for Intersection I

Initialize queue lengths $Q \leftarrow \mathbf{0}$, phase u, min green time per phase g_u, yellow time y, current phase $\overline{u}$, switch threshold s, switch time $h_s \leftarrow \infty$
Define $\mathcal{U}$ as set of phases and $\mathcal{N}_u$ as set of phases which share a direction with u
for *each time* h **do**
    if $h - h_s > g_u$ **then**
        Calculate weights in each lane w_ℓ
        Calculate weights in each phase w_u
        if $\max_{u \in \mathcal{U}} w_u - w_{\overline{u}} > \eta w_{\overline{u}}$ **then**
            $\overline{u} \leftarrow \operatorname{argmax}_{u \in \mathcal{U}} w_u$
            $Z \leftarrow 1$
            $t_s \leftarrow t$
        else if $\max_{\tilde{u} \in \mathcal{N}_u} w_{\tilde{u}} - w_{\overline{u}} > 0.5\eta w_{\overline{u}}$ **then**
            $\overline{u} \leftarrow \operatorname{argmax}_{\tilde{u} \in \mathcal{N}_u} w_{\tilde{u}}$
            $Z \leftarrow 1$
            $t_s \leftarrow t$
        end
        if $Z = 1$ **then**
            Send packet of data D containing source intersection I, Q_ℓ for all lanes ℓ leading to neighboring intersection J, and expected time of arrival T
            $Z = 0$
        end
    end
    $h \leftarrow h + \Delta h$
end

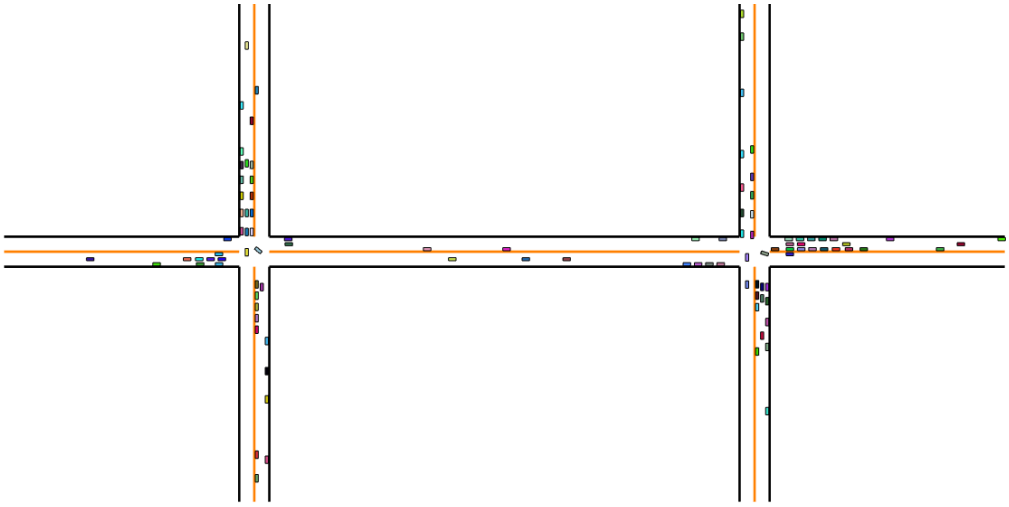


Figure 4.2: Image of microscopic model.

Simulations were run on single intersections and systems of two intersections, see Figure 4.2. We define the discrete time step $h = 0.1$. The safety gap is $t_{\text{gap}} = 1$ second, so a vehicle maintains a gap with the vehicle in front such that they would not collide even with a one second delay in reaction to the vehicle ahead. The speed limit is $v_f = 20$ kph, yellow time is $y = 5$ seconds for each phase. The minimum time in green was initially 10 seconds, however during simulation we found this to be inefficient, so an adaptive minimum green time was used, where $g_p^I = \min\{10, 3.5 + 1.5 \max_{\ell \in L} Q_\ell^I\}$, where L is the set of lanes which are allowed through intersection I in phase p, Q_ℓ^I is the length of the queue in lane ℓ, and the coefficients were chosen to give enough time for each vehicle to pass through. Travel time across intersections $T_p = 17.5$. The scaling factor in Equation (4.5) is $\phi = .05$, the switch threshold is $\eta = 2$, and the wait threshold is $\gamma = 0.1$ as defined in Section 4.3. Vehicles spawn at each time step according to a Poisson distribution with parameter λ, and vehicles are distributed into available spawn points with uniform probability unless stated otherwise. The spawn rate was chosen by selecting the highest value in which queues remained bounded for all time, this value was around $\lambda \approx 0.8$ veh/sec, see Figure 4.3.

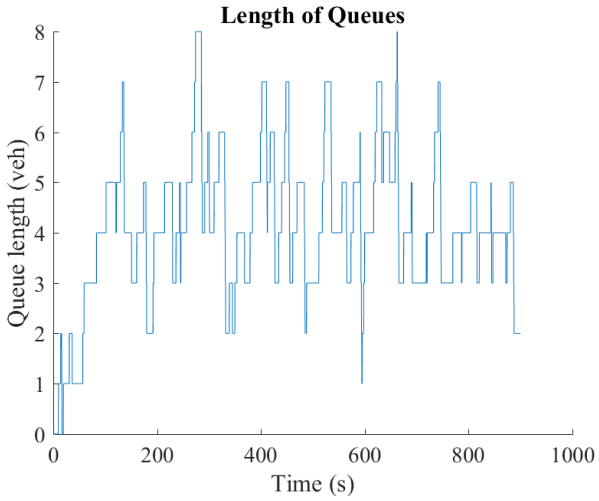


Figure 4.3: Plot of max queue length

In the single intersection case, our intersection algorithm significantly outperforms a simple intersection with constant phase cycles, where an entire cycle takes 1 minute. We use three performance metrics to compare algorithms, the total travel time of all vehicles *tt*, the total wait-time of all vehicles *twt*, and the total weighted wait-time of all vehicles *twwt*. It can be seen in Figures 4.4 and 4.5 that the wait-time is kept lower using our algorithm. Results of a few five minute simulations on one intersection are found in Table 4.6.

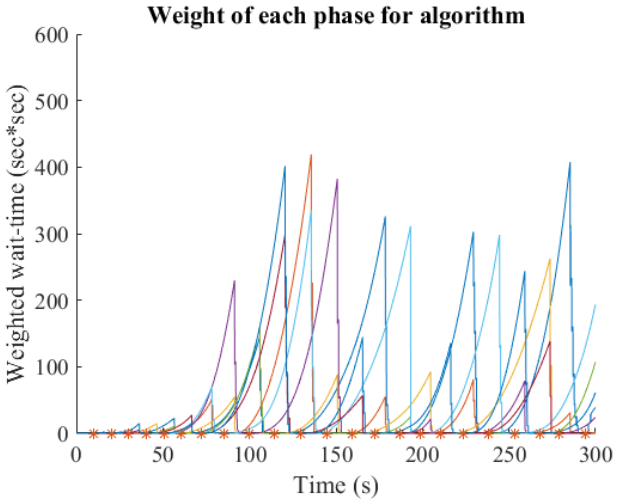


Figure 4.4: Evolution of weight per phase under custom algorithm

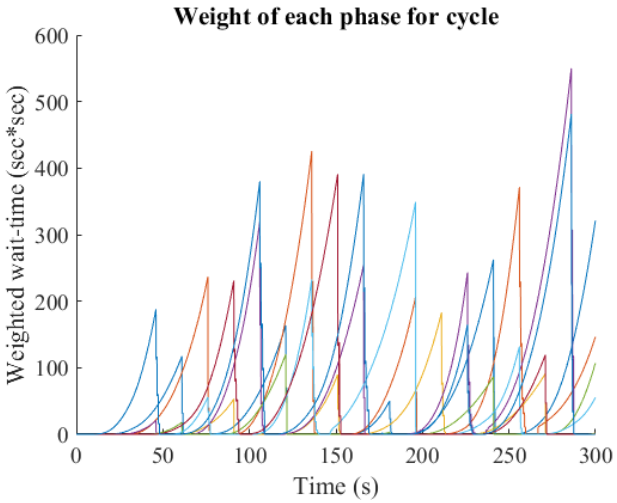


Figure 4.5: Evolution of weight per phase under cyclical algorithm

	Cycle			Custom		
Ω	*tt*	*twt*	*twwt*	*tt*	*twt*	*twwt*
0.60	5.88	3.26	5.32	5.00	2.29	2.83
0.80	8.20	4.50	7.83	7.42	3.96	6.33
1.00	12.00	8.38	18.8	10.40	5.78	10.30

Figure 4.6: Performance of algorithms, in kiloseconds

In the two intersection case simulated over five minutes, we compared each performance given different values of α using Algorithm 7, where α is the coefficient on the cooperation term B_ℓ, see Figure 4.7. This simulation used a non-uniform Poisson spawn, where vehicles are twice as likely to spawn on the road which passes through both intersections. These simulations were run for two minutes each. We can see a trade-off where increasing α reduces *tt* but increases *twt* and *twwt*, which is expected because B_ℓ is perturbing an algorithm which is minimizing *twwt* to allow more vehicles through immediately. In Figure 4.8, five minute simulations were run to better characterize performance as a function of α when α is small. The thin lines represent the performance under the default cyclic policy under the same conditions. Notice the large decrease in twwt but small increase in tt by using our algorithm.

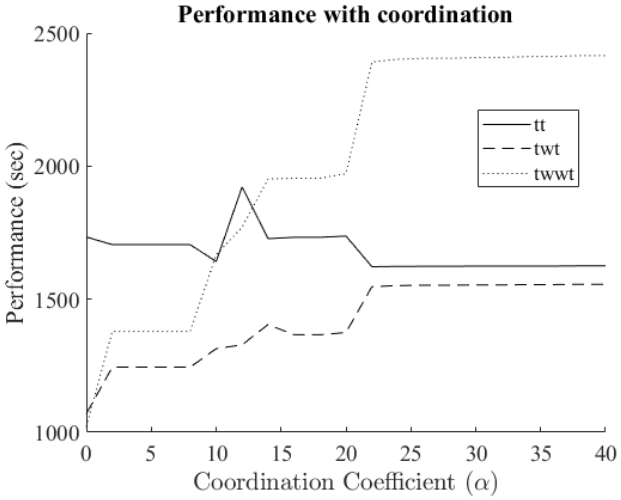


Figure 4.7: Plot of performance over coordination coefficient

4.6 Conclusion and Future Work

In summary, we presented a novel intersection control algorithm based on an objective function which attempts to accurately characterize driver preferences. A realistic vehicle model is constructed and a simplified version is used to guarantee bounded weighted queues given small spawn rates. The algorithm is tested thoroughly in simulation and compared with other methods.

We are considering several possible extensions to this work. One potential problem with this formulation is that vehicles must be delayed before they are considered important, perhaps a more sophisticated model not based on a threshold value will perform better. We plan to compare performance to more established algorithms such as SCOOT or SCATS. We are interested in addressing the spillover problem, when the queue from one intersection spills into another, we believe the current formulation would allow this to be addressed through an additional term in the objective function. We are also considering a learning-based method for tuning parameters such as wait threshold γ, switch threshold η, time spent in each phase g_u^I, and coordination coefficient α.

Chapter 4, in part, is submitted for publication to IEEE Conf. on Control Tech-

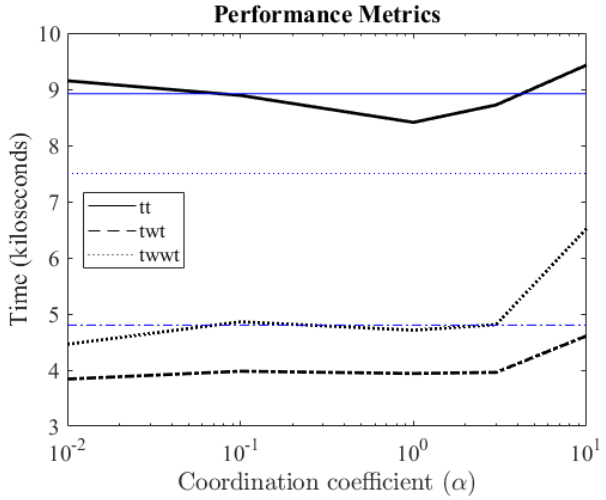


Figure 4.8: Zoomed plot of performance over coordination coefficient

nology and Applications, 2017, by E. Gravelle and S. Martínez and IET Control Theory & Applications, 2017, by E. Gravelle and S. Martínez. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Closing Remarks

In this dissertation, we have made a few contributions to the vast and ever-growing field of distributed systems. We have formulated problems with specific objectives for load balancing, lane reversal, rerouting, and intersection control, mathematically modeled the systems, analyzed their stability and convergence properties, and tested their performance in simulation.

There is much future work involving both more analysis and more experiments. Analyzing each system's stability under uncertainty, various noise conditions, perturbations, and modeling errors would show the systems robustness properties in greater detail. Convergence rates should be characterized or bounded to provide realistic time scales for other decisions.

On the other hand, increasing the complexity of the models to come closer and closer to real world systems would make the simulations better predictors of true performance. While analysis may become infeasible with a very complex model, the behavior of the simulated system could be used to discover and test solutions for many complications that may not arise in the simpler model.

Over time, more and more responsibility is put into computers as they continue to

expand they domains in which they outperform humans. In this context, this dissertation has attempted to take a step towards establishing intelligent distributed algorithms that improve efficiency of various systems while ensuring overall stability.

Bibliography

[1] A. Abadi, P. A. Ioannou, and M. M. Dessouky. Multimodal dynamic freight load balancing. *IEEE Transactions on Intelligent Transportation Systems*, 2015. Scheduled for publication.

[2] T. C. Au, N. Shahidi, and P. Stone. Enforcing liveness in autonomous traffic management. In *AAAI Conference on Artificial Intelligence*, pages 1317–1322, 2011.

[3] T. C. Aysal, M. J. Coates, and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56(10):4905–4918, 2008.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[5] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 2 edition, 1994.

[6] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *IEEE Int. Conf. on Decision and Control and European Control Conference*, pages 2996–3000, Seville, Spain, December 2005.

[7] G. Bretti, R. Natalini, and B. Piccoli. Numerical approximations of a traffic flow model on networks. *Networks and Heterogeneous Media*, 1(1):57–84, 2006.

[8] M. Campbell, M. Egersdedt, J. P. How, and R. M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A*, 368(1928):4649–4672, October 2010.

[9] R. Carli and S. Zampieri. *Advances in Control Theory and Applications*. Springer Berlin Heidelberg, 2007.

[10] D. Carlino, S. D. Boyles, and P. Stone. Auction-based autonomous intersection management. In *IEEE Int. Conf. on Intelligent Transportation Systems*, pages 529–534, 2013.

[11] Y. C. Chiu, H. Zheng, J. A. Villalobos, W. Peacock, and R. Henk. Evaluating regional contra-flow and phased evacuation strategies for texas using a large-scale dynamic traffic simulation and assignment approach. *Journal of Homeland Security and Emergency Management*, 5(1), 2008.

[12] C. F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[13] C. Diakaki, M. Papageorgiou, and K. Aboudolas. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, 10(2):183–195, 2002.

[14] D. Downey, 2008. http://www.sandiegouniontribune.com/sdut-region-caltrans-rolls-out-barrier-moving-machine-2008aug18-story.html.

[15] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

[16] Y. Dujardin, F. Boillot, D. Vanderpooten, and P. Vinant. Multiobjective and multimodal adaptive traffic light control on single junctions. In *IEEE Int. Conf. on Intelligent Transportation Systems*, pages 1361–1368. IEEE, 2011.

[17] M. P. Fanti, M. Franceschelli, A. M. Mangini, G. Pedroncelli, and W. Ukovich. Discrete consensus in networks with constrained capacity. In *IEEE Int. Conf. on Decision and Control*, pages 2012–2017, 2013.

[18] M. Franceschelli, A. Giua, and C. Seatzu. Load balancing on networks with gossip-based distributed algorithms. In *IEEE Int. Conf. on Decision and Control*, pages 500–505, 2007.

[19] M. Franceschelli, A. Giua, and C. Seatzu. Load balancing over heterogeneous networks with gossip-based algorithms. In *American Control Conference*, pages 1987–1993, 2009.

[20] M. Franceschelli, A. Giua, and C. Seatzu. A gossip-based algorithm for discrete consensus over heterogeneous networks. *IEEE Transactions on Automatic Control*, 55(5):1244–1249, 2010.

[21] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri. Average consensus by gossip algorithms with quantized communication. In *IEEE Int. Conf. on Decision and Control*, pages 4831–4836, Cancún, México, December 2008.

[22] M. Hausknecht, T. C. Au, and P. Stone. Autonomous intersection management: Multi-intersection optimization. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 4581–4586, 2011.

[23] M. Hausknecht, T. C. Au, and P. Stone. Dynamic lane reversal in traffic management. In *IEEE Int. Conf. on Intelligent Transportation Systems*, pages 1929–1934, 2011.

[24] P. Hunt, D. Robertson, R. Bretherton, and R. Winton. SCOOT - a traffic responsive method of coordinating signals. Technical report, 1981.

[25] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[26] A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

[27] C. K. Keong. The glide system - singapore's urban traffic control system. *Transport Review*, 13(4):295–305, 1993.

[28] D. Laibson. Life-cycle consumption and hyperbolic discount functions. *European Economic Review*, 42(3):861–871, 1998.

[29] J. Lavaei and R. M. Murray. Quantized consensus by means of gossip algorithm. *IEEE Transactions on Automatic Control*, 57(1):19–32, 2012.

[30] M.J. Lighthill and G.B. Whitham. On kinematic waves. II. a theory of traffic flow on long crowded roads. *Royal Society of London. Proceedings Series A: Mathematical, Physical and Engineering Sciences*, 229(1178):317–345, 1955.

[31] P. Lowrie. The sydney coordinated adaptive traffic system-principles, methodology, algorithms. In *Int. Conference on Road Traffic Signalling*, number 207, 1982.

[32] Q. Meng, H. Khoo, and R. Cheu. Research on reversal lane application method of urban road network based on the bi-level programming. *Advances in Intelligent Systems and Computing*, 279:983–992, 2014.

[33] A. Nedić, A. Olshevsky, and A. Ozdaglar. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[34] R. Noland and K. Small. *Travel-time uncertainty, departure time choice, and the cost of the morning commute*. Institute of Transportation Studies, University of California, Irvine, 1995.

[35] G. De Nunzio, G. Gomes, C.C. de Wit, R. Horowitz, and P. Moulin. Arterial bandwidth maximization via signal offsets and variable speed limits control. In *IEEE Int. Conf. on Decision and Control*, pages 5142–5148. IEEE, 2015.

[36] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[37] E. Panteley and A. Loria. Synchronization and dynamic consensus of heterogeneous networked systems. *IEEE Transactions on Automatic Control*, PP, 2017.

[38] H. Payne and S. Thompson. Malfunction detection and data repair for induction-loop sensors using i-880 data base. *Transportation Research Record: Journal of the Transportation Research Board*, (1570):191–201, 1997.

[39] R. Rajagopal and P. Varaiya. Evaluating the health of californias loop sensor network. *University of California, Berkeley*, 94720, 2007.

[40] P. I. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956.

[41] Z. Shen, K. Wang, and F. Zhu. Agent-based traffic simulation and traffic signal timing optimization with gpu. In *IEEE Int. Conf. on Intelligent Transportation Systems*, pages 145–150. IEEE, 2011.

[42] P. Tallapragada and J. Cortés. Coordinated intersection traffic management. *IFAC-PapersOnLine*, 48(22):233–239, 2015. *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Philadelphia, PA.

[43] Z. Tian and T. Urbanik. System partition technique to improve signal coordination and traffic progression. *Journal of Transportation Engineering*, 133(2):119–128, 2007.

[44] J. W. Wang, H. F. Wang, W. J. Zhang, W. H. Ip, and K. Furata. Evacuation planning based on the contraflow technique with consideration of evacuation priorities and traffic setup time. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):480–485, 2012.

[45] J. T. Wen and M. Arcak. A unifying passivity framework for network flow control. *IEEE Transactions on Automatic Control*, 49(2):162–174, 2004.

[46] T. Wongpiromsarn, T. Uthaicharoenpong, Y. Wang, E. Frazzoli, and D. Wang. Distributed traffic signal control for maximum network throughput. In *IEEE Int. Conf. on Intelligent Transportation Systems*, pages 588–595. IEEE, 2012.

[47] D. B. Work, S. Blandin, O. P. Tossavainen, and A. M. Bayen. Applied mathematics research express (amrx). *Applied Mathematics Research eXpress (AMRX)*, pages 1–35, 2010.

[48] C. Wuthishuwong and A. Traechtler. Coordination of multiple autonomous intersections by using local neighborhood information. In *Int. Conf. on Connected Vehicles and Expo*, pages 48–53, 2013.

[49] D. Xue and Z. Dong. An intelligent contraflow control method for real-time optimal traffic scheduling using artificial neural network, fuzzy pattern recognition, and optimization. *IEEE Transactions on Control Systems Technology*, 8(1):183–191, 2000.

[50] W. W. Zhou, P. Livolsi, E. Miska, H. Zhang, J. Wu, and D. Yang. An intelligent traffic responsive contraflow lane control system. In *Vehicle Navigation and Information Systems Conference*, pages 174–181, 1993.

[51] M. Zhu and S. Martínez. On the convergence time of asynchronous distributed quantized averaging algorithms. *IEEE Transactions on Automatic Control*, 56(2):386 – 390, 2011.