DALI: A COLLABORATIVE, AGENT-BASED

TRAFFIC SIGNAL TIMING SYSTEM

by

Behnam Torabi

APPROVED BY SUPERVISORY COMMITTEE:

Rym Zalila-Wenkstern, Chair

Farokh Bastani

Lawrence Chung

Kang Zhang

This work is dedicated to my parents and sisters,

for their endless love, encouragement, and support.

DALI: A COLLABORATIVE, AGENT-BASED TRAFFIC SIGNAL TIMING SYSTEM

by

BEHNAM TORABI, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

SOFTWARE ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS

May 2019

ACKNOWLEDGMENTS

I would like to express my most profound appreciation to my supervising Professor, Dr. Rym Zalila-Wenkstern, who has shown me, by her example, what a professional scientist is. She has not only been my mentor but also family and a good friend to me. I cannot express enough thanks to my dissertation committee, Professor Farokh Bastani, Professor Lawrence Chung, and Professor Khang Zhang for their constructive feedback and expertise, and for devoting their precious time and effort to serve on my proposal and dissertation defense. I would also like to express my gratitude to my friends who made this challenging journey pleasant for me. Finally, I want to thank Mr. Robert Saylor, whose valuable cooperation enabled me to advance my research.

March 2019

DALI: A COLLABORATIVE, AGENT-BASED

TRAFFIC SIGNAL TIMING SYSTEM

Behnam Torabi, PhD
The University of Texas at Dallas, 2019

Supervising Professor: Rym Zalila-Wenkstern, Chair

In this dissertation, we present DALI (Distributed, Agent-based traffic LIghts), a smart collaborative traffic signal timing system. With DALI, intersection controller agents communicate with each other through direct links and do not have a supervising unit to oversee coordination. By default, they execute a timing strategy that improves traffic flow. At the same time, they observe and analyze their respective intersections. If, at any given time, an agent determines that its intersection is congested, it deliberates and defines a new timing plan. It also determines which direct intersections may be affected by the new timing plan and communicates with the concerned intersection agents. They in turn communicate with those agents that may be affected, and the process continues until all affected intersections are notified. The agents then negotiate and collaborate with one another to ensure that the traffic flow will be optimized throughout the intersections. DALI was validated by traffic engineers as well as through extensive simulation of the City of Richardson's traffic network. In addition, hybrid simulations (i.e., integration of controllers in the field with the simulator) were run to verify compliance with the strict traffic regulations. DALI was deployed in a Richardson, Texas, corridor that includes three major intersections. The data collected for a period of three weeks shows that in average, DALI reduced delay by 40.12 percent (43.56 percent during weekday peak hours).

TABLE OF CONTENTS

ACKNOWLEDGMENTS v
ABSTRACT vi
LIST OF FIGURES xi
LIST OF TABLES xiii
INTRODUCTION 1
CHAPTER 1 BACKGROUND 4
1.1 Agents and Multi-Agent Systems 4
1.2 Traffic Concepts 4
1.2.1 Components of a Traffic Control System 4
1.2.2 Basic Signal Timing Parameters 7
1.2.3 Signal Operation Modes 8
1.2.4 Traffic Signal Coordination 9
1.3 City Of Richardson 10
1.3.1 Intelight Controllers 10
1.3.2 Traffic Management Center 11
1.3.3 Operation Modes 11
1.3.4 Detection System 12
1.3.5 Defining Timing Plans 13
1.4 Related Works 13
1.4.1 Optimizations at Network or Sub-Network Level 14
1.4.2 Optimizations at Intersection Level 16
CHAPTER 2 DALI'S ALGORITHMS 22
2.1 DALI Model 22
2.2 Model Definitions 23
2.2.1 Set Definitions 23
2.2.2 Function Definitions 24
2.3 Non Congested Traffic Conditions 25
2.3.1 Communicating Vehicle Arrivals 26

2.3.2 Queue Length Estimation . . . 26
2.3.3 Evaluating Timing Plans . . . 27
2.4 Congested Traffic Conditions . . . 28
2.4.1 Detecting Congestion . . . 30
2.4.2 Defining new configuration . . . 31
2.4.3 Requesting Agents' Feedback . . . 32
2.4.4 Computing Level Of Agreement . . . 32
2.4.5 Special Cases . . . 34
2.5 Adaptive Assignment of Threshold Values . . . 35
2.5.1 Overview of Reinforcement Learning . . . 35
2.5.2 Agent Algorithms for Adaptive Threshold Assignment . . . 36
CHAPTER 3 CASE STUDY . . . 39
CHAPTER 4 MATISSE: A SIMULATION SYSTEM FOR AGENT-BASED INTELLIGENT TRANSPORTATION SYSTEMS . . . 42
4.1 Overview of MATISSE 2.0 . . . 42
4.1.1 High Level Architecture . . . 42
4.1.2 Traffic Network Structure . . . 43
4.1.3 Creating Virtual Agents . . . 43
4.2 MATISSE 3.0 . . . 45
4.2.1 Importing Traffic Networks from OpenStreetMap . . . 45
4.2.2 Defining Vehicle Distribution at Initialization Time . . . 48
4.2.3 Defining New Vehicle and Intersection Controller Behaviors . . . 48
4.2.4 Hybrid Simulation . . . 50
CHAPTER 5 EVALUATION OF DALI'S SIMULATION RESULTS . . . 51
5.1 Metrics . . . 51
5.2 Simulation Setting . . . 51
5.3 Experiment 1: DALI with fixed threshold values . . . 52
5.3.1 Experiment 1.1: Normal Traffic Conditions . . . 53
5.3.2 Experiment 1.2: Normal Traffic Conditions With Accident . . . 54

5.3.3 Experiment 1.3: Continuous Random Traffic Conditions 55
5.3.4 Experiment 1.4: Continuous Random Traffic Conditions with Accident 56
5.4 Experiment 2: DALI with Adaptive Threshold Values 57
5.4.1 Experiment 2.1: Assessing Delay 57
5.4.2 Experiment 2.2: Assessing Changing Values of Threshold a 58
5.4.3 Experiment 2.3: Assessing Changing Values of Threshold g 58
5.5 Experiment 3: Hybrid Simulation 60
CHAPTER 6 DEPLOYMENT OF DALI 64
6.1 The Waterview Corridor 64
6.2 Safety and Monitoring Requirements 64
6.3 Agent Implementation 65
6.4 Intelight Controllers 67
6.5 Agent-Controller Interaction Mechanism 70
6.6 Executing DALI Agents 71
6.7 Lessons Learned 71
CHAPTER 7 EVALUATION OF DALI'S DEPLOYMENT RESULTS 73
7.1 Metrics and Data Collection 73
7.2 Queue Length 73
7.3 Delay 76
7.4 Cost of Delay 76
CONCLUSION 79
7.5 Contributions 79
7.5.1 Model and Algorithm Definitions 79
7.5.2 Traffic Simulator Feature Development 79
7.5.3 Deployment 80
7.6 Future Work 80
7.6.1 Model and Algorithm Definitions 81
7.6.2 Traffic Simulator Feature Development 81
7.6.3 Deployment 81

REFERENCES 83
BIOGRAPHICAL SKETCH 90
CURRICULUM VITAE 91

LIST OF FIGURES

1.1 Overview Of An Intersection Control System. 5
1.2 Overview Of A Traffic Management System. 5
1.3 Standard Phase Numbers and Induction Loops Area. 7
1.4 Timing Diagram For An intersection With Three Phases 7
1.5 Coordination Of Traffic Signals 10
1.6 City of Richardson's roads map. 11
1.7 Intelight Cabinet and Controller (Intelight, 2019). 12
1.8 Induction Loops in City of Richardson. 12
2.1 c_n determines $rateOut$ and receives $rateIn$ from c_m. 29
2.2 $PercentCong$ of phase $ph_{c_n,k}$. 31
3.1 Overview of the network in the case study. 40
3.2 Intersection assigned to c_2. 40
4.1 Traffic Network Definition in MATISSE 43
4.2 Code for vehicle Agent 45
4.3 Generate an intersection from an OSM file 46
4.4 An Intersection in Paris and Its Representation in Open Street Map 47
4.5 Vehicle agents perception. a) Circle of Influence b) Vision cones 48
4.6 Four possible actions of a vehicle 49
4.7 Architecture of the Hybrid Simulation. 50
5.1 2D visualization of Richardson's Traffic Network. 52
5.2 Average delay using traffic data from the City of Richardson 53
5.3 Average delay with accident in peak morning hours using real traffic data 54
5.4 Average delay for random traffic patterns 55
5.5 Average delay for random traffic patterns with accidents. 56
5.6 Average delay using traffic data from the City of Richardson 58
5.7 Average Delay For Different Values of a. 59
5.8 Average delay For Different Values of g. 60
5.9 Average Group Size For Different Values of g. 61

5.10 Arterial image of simulated intersections. 62
5.11 Average incoming traffic flow at different times of work days. 62
5.12 Delay Reduction For different traffic flows at different entrances. 63
6.1 Overview of The Corridor. 65
6.2 Overview of Intersections. 66
6.3 DALI Agents Running in the Lab. 67
6.4 a) Status of the intersection in MAXTIME. b) Actions in MAXTIME 67
6.5 a) A day plan in MAXTIME. b) Patterns in MAXTIME 68
6.6 a) The schedule table in MAXTIME. b) Schedule rule for DALI 69
6.7 a) Assignments of input points in MAXTIME. b) Assignments of virtual detectors for DALI. 69
6.8 a) Status of input points in MAXTIME b) Status of detectors in MAXTIME. . 70
6.9 a) Dayplan for DALI. b) Pattern for DALI 70
7.1 Traffic Flow Rate for Different Times of the Day. 74
7.2 Queue Length Reduction. 75
7.3 Delay Reduction. 77
7.4 Average delay and cost of the delay for each vehicle at each intersection. 78

LIST OF TABLES

5.1 Number of Signalized Intersection with various incoming and outgoing lanes . . 52
5.2 Number of Non-Signalized Intersection with various incoming and outgoing lanes 52
5.3 Number of Message Exchanges For Different Values of a. 59
5.4 Reduction in Delay for Different Traffic Flows 63
7.1 Distribution of data collection time 74

INTRODUCTION

Traffic signals impact virtually everyone every day. Whether on congested or uncongested routes, traffic signals punctuate every urban trip and have a direct impact on drivers, the environment, and the economy (Day et al., 2010).

Several Traffic Signal Timing systems (TST) have been proposed by manufacturers, traffic engineers and researchers. The purpose of a TST is to coordinate individual traffic signals to achieve network-wide traffic operational objectives. A TST usually consists of several components: a) a number of intersection controllers, i.e., devices which control the operations of the intersection's traffic signals; b) a communication network and c) either a central computer or a hierarchy of computers to manage the system. Coordination is implemented through a number of techniques including time-based and hardwired interconnection methods.

Modern TSTs rely upon the detection of traffic conditions in real-time to determine effective signal settings. Generally, conventional TSTs define the traffic signal timing problem as the optimization of a set of signal timing parameters (e.g., split, cycle length, offset) for an objective function (e.g., minimizing delay, minimizing travel time, maximizing traffic flow). Many conventional TSTs have been proposed by traffic engineers and researchers. *Fully centralized* TSTs (Hunt et al., 1982; Sims and Dobinson, 1980) allow for efficient coordination of intersection controllers under normal traffic conditions but do not perform well when major traffic disruptions occur. *Partially centralized* TSTs (Mirchandani and Head, 2001; Gartner, 1982) adapt to certain traffic variations within fixed constraints, but require knowledge that is difficult to obtain in practice. *Decentralized* TSTs respond quickly to any traffic demand by generating unconstrained signal timings (Henry et al., 1983) but use complex optimization algorithms which severely limit their scalability.

The application of the agent paradigm to traffic signal timing has been of interest to Multi-Agent System (MAS) researchers for some time. Distribution, autonomy and coordination are agent properties that are naturally suited for the traffic domain. In the context of traffic signal timing, researchers have proposed the use of a variety of techniques including game theory (Bazzan, 2005; Cheng et al., 2006), neural networks (Srinivasan et al., 2006; Chao et al., 2008), fuzzy logic (Collotta et al., 2015; Bi et al., 2014) as well as the commonly used Reinforcement Learning (RL). RL-based solutions attempt to address two types of traffic signal timing problems: non-coordinated and coordinated. In non-coordinated RL-systems, an agent's goal is to optimize the signal timing at its intersections only. The lack of coordination between agents often leads to a degradation of the overall traffic conditions. On the other hand, in coordinated agent systems, agents implicitly coordinate with their direct neighbors by sharing their states and intended actions. Given the astronomical number of states and actions that need to be considered for any realistic traffic model, coordinated RL systems have no option but to overly simplify the traffic model. Other agent-based systems using vehicle-to-vehicle and vehicle-to-infrastructure (V2X) communications have been proposed (Younes and Boukerche, 2016; Dresner and Stone, 2008). Although some of these approaches provide impressive simulation results (Dresner and Stone, 2008), they are based on assumptions that do not have their counterparts in the real world. In addition, V2X communication technologies are still in their infancy and their global deployment is years away.

In this dissertation, we present DALI (Distributed Agent-based traffic LIghts), a collaborative multi-agent traffic signal timing system for congestion reduction. In DALI, intersection controllers are augmented with agents which communicate with one another through direct links. The agents collaboratively adapt signal timings by considering the feedback of all agents affected by a change. DALI was deployed in the City of Richardson's Waterview Parkway corridor at three major intersections. The data collected for a three week period shows that on average, DALI reduced delay by 40.12%.

This dissertation is organized as follows: Chapter 1 gives the background knowledge for the topic of our research and discusses related works. Chapter 2 presents the DALI's model and algorithms. Chapter 3 illustrates the DALI's operations for congestion reduction through a detailed case study. Chapter 4 gives an overview of MATISSE (Al-Zinati and Zalila-Wenkstern, 2015), a multi-agent traffic simulation system and discuss the new features that were developed for the purpose of this research. Chapter 5 presents experimental results in simulated and hybrid environments. Chapter 6 discusses the deployment of DALI in the City of Richardson and Chapter 7 evaluates DALI's performance in a real-world setting. We conclude by summarizing our contributions and discussing future work.

CHAPTER 1

BACKGROUND

In this chapter we review some background information, starting with definitions of important terms used in the field of multi-agent based traffic management systems. Then, we discuss City of Richardson's traffic management system and review existing works.

1.1 Agents and Multi-Agent Systems

Agent: A software entity that is situated in an environment, and that is capable of autonomous action within this environment in order to meet its design objectives (Wooldridge, 2009).

Multi-Agent System (MAS): A system that consists of a number of agents, which interact with one another by exchanging messages. In order to successfully interact, agents require the ability to cooperate, coordinate, and negotiate with each other (Wooldridge, 2009). MAS is responsible for hosting such interaction.

1.2 Traffic Concepts

1.2.1 Components of a Traffic Control System

Traffic signal: A signalling device placed along, beside, or above a roadway to guide, warn, and regulate the flow of traffic.

Traffic signal controller: An electrical device mounted in a *cabinet* at the intersection that controls the operation of traffic signals. In this dissertation, we refer to traffic signal controllers as **intersection controllers**.

Detection system: A system for indicating the presence or passage of vehicles using sensors. Examples of sensors include magnetometers, which may be placed underneath a paved roadway or bridge structure; video image processors that use cameras; microwave radar;

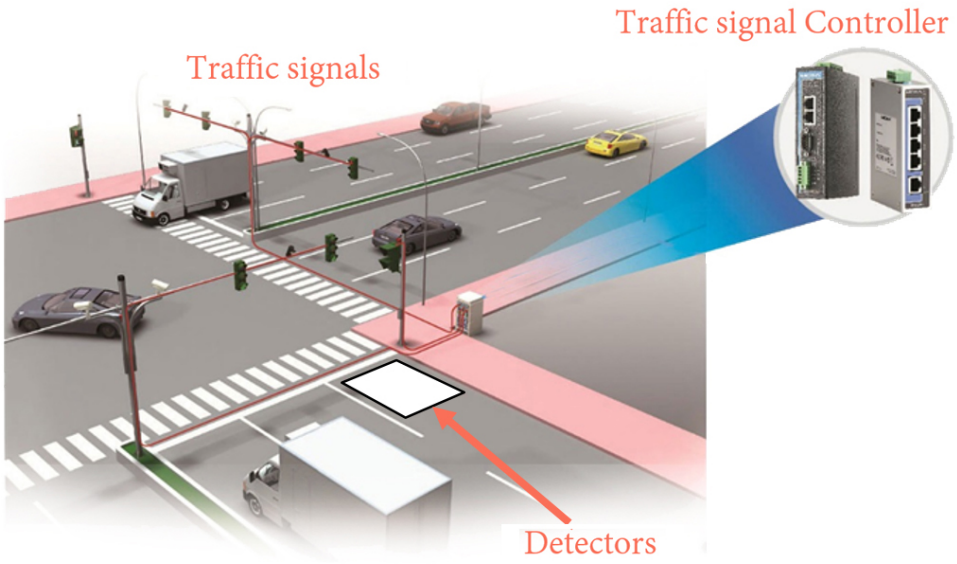

Figure 1.1. Overview Of An Intersection Control System.

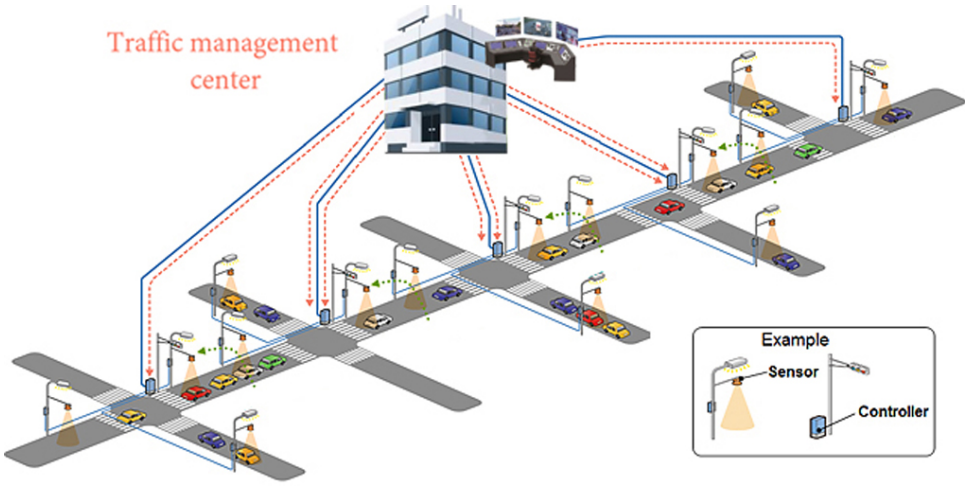

Figure 1.2. Overview Of A Traffic Management System.

ultrasonic, and passive infrared sensors installed on tall poles next to the roadway or traffic signal mast arms; laser radar sensors installed on structures that span the lanes to be monitored and *inductive loop detectors*, which are sawcut into the pavement.

Inductive loop detectors: A detection system that responds to the presence of vehicles on the roadway by relying upon the impact of the conductive mass of the vehicle on the alternating magnetic field of a wire loop. When a vehicle passes over the loop of wire mounted under the surface of the roadway, it reacts with the alternating magnetic field that is associated with that loop. On standard loops, this reaction is a reduction in loop inductance. The decreased inductance actuates an electronic unit which sends a pulse to the intersection controller signifying the passage or presence of a vehicle.

Traffic management centers (TMCs): A mission control for an urban area's major arterials and highway network. A TMC monitors traffic signals, intersections, and roads and proactively deploys traffic management strategies to decrease congestion and coordinate authorities during emergencies and special events. TMCs gather online data through continuous communication with intersection controllers. An overview of an intersection control system can be seen in Figure 1.1.

Communications: Wire-line communications have been regularly used to transfer information between a traffic signal controller and a TMC or other traffic signal controllers. While this alternative remains, additional options have become available in the last few years. These include closed-circuit televisions (CCTV), variable message signs, short range and long range wireless ethernet.

The components discussed above form a Traffic Control System (TCS). TCSs may have different architectures and use various approaches to enhance traffic flow in urban areas. An overview of a traffic management system can be seen in Figure 1.2.

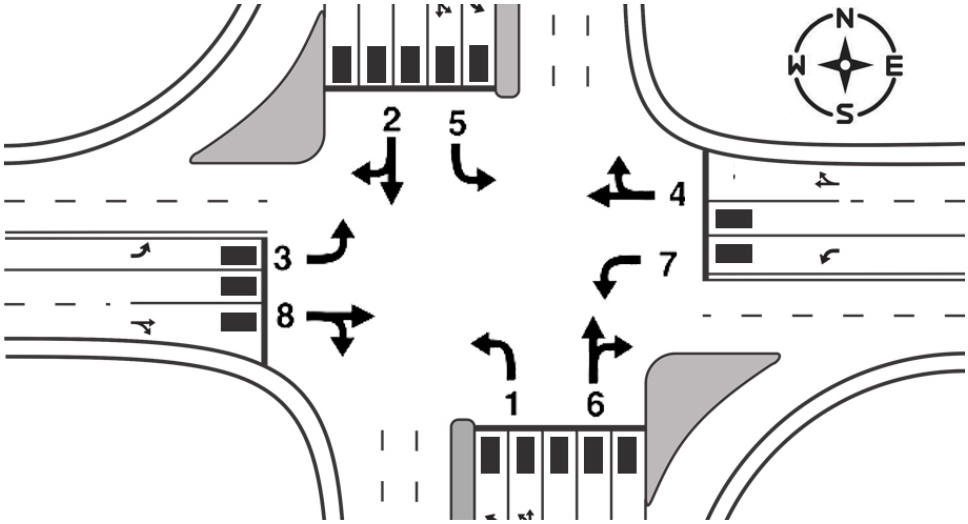

Figure 1.3. Standard Phase Numbers and Induction Loops Area.

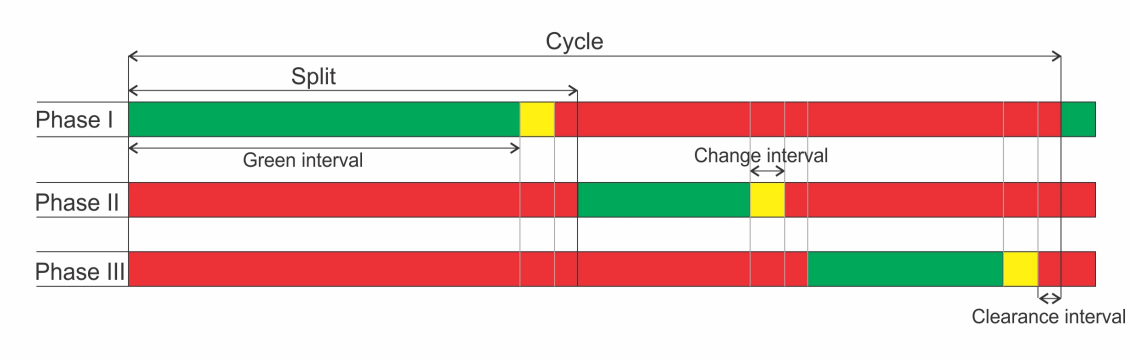

Figure 1.4. Timing Diagram For An intersection With Three Phases

1.2.2 Basic Signal Timing Parameters

The definitions given in this section are borrowed from the United States Department of Transportation manual for traffic signal timing (Koonce et al., 2008).

Movement: A term used to describe a vehicle's action (e.g., turning or going straight) at an intersection.

Phase: A group of non-conflicting movements receiving the same signal at the same time. Standard controllers provide eight phases to serve a standard four-legged intersection (see Figure 1.3).

Interval: A period during which traffic signals do not change.

Vehicular green interval: A time devoted to serving a vehicular phase with a green signal.

Change interval: A yellow signal aiming to warn drivers of the impending change in green interval assignment.

Clearance interval: A duration in which all phases are red to provide additional time before the time when the next phase becomes green.

Minimum green: A time which represents the least amount of time that a green interval can last.

Maximum green: A time which represents the maximum amount of time that a green interval can last when there is a demand for a conflicting phase.

Split: The time assigned to a phase to be green plus change and clearance intervals (see Figure 1.4).

Offset: A parameter determining the start and/or the end of a split.

Cycle Length: This is the total time to complete one sequence of signalization around an intersection.

1.2.3 Signal Operation Modes

Traffic signals operate in either *pre-timed* or *actuated* mode or some combination of the two.

Pre-timed Control

In *pre-timed mode*, intervals are pre-set according to a predetermined schedule, based on historical traffic patterns. Pre-timed control is ideally suited for closely spaced intersections where traffic volumes and patterns are consistent on a daily or day-of-week basis, (e.g., downtown areas). They are also better suited for intersections where three or fewer phases are needed.

Actuated Control

Actuated control consists of intervals that are called and extended in response to vehicle detectors. The duration of each phase is determined by detector input and corresponding controller parameters. Actuated control can be characterized as *fully-actuated* or *semi-actuated*, depending on the number of traffic movements that have detectors.

Semi-Actuated Control

Semi-actuated control uses detection only for the minor movements at an intersection. The phases associated with the major-road through movements are operated as *non-actuated*. In this type of operation, the controller is programmed to dwell in the non-actuated phase and, thereby, sustain a green indication for the highest flow movements (normally the major street through movement). Minor movement phases are served after either the presence of a vehicle is detected or the major movement phase reaches its maximum green time. Controllers that operate in semi-actuated mode are suitable for isolated intersections with a low-speed major road and lighter crossroad volume.

Fully-Actuated Control

Fully-actuated control refers to intersections for which all phases are actuated and, hence, it requires detection for all traffic movements. Fully-actuated control is ideally suited to isolated intersections where the traffic demands and patterns vary widely during the course of the day.

1.2.4 Traffic Signal Coordination

To minimize traffic delay, it is desirable that a platoon of vehicles leaving one intersection arrives at the next intersection during a green interval. This is achieved by *coordinating* the operations of adjacent signals. Coordination of traffic signals occurs when a group of

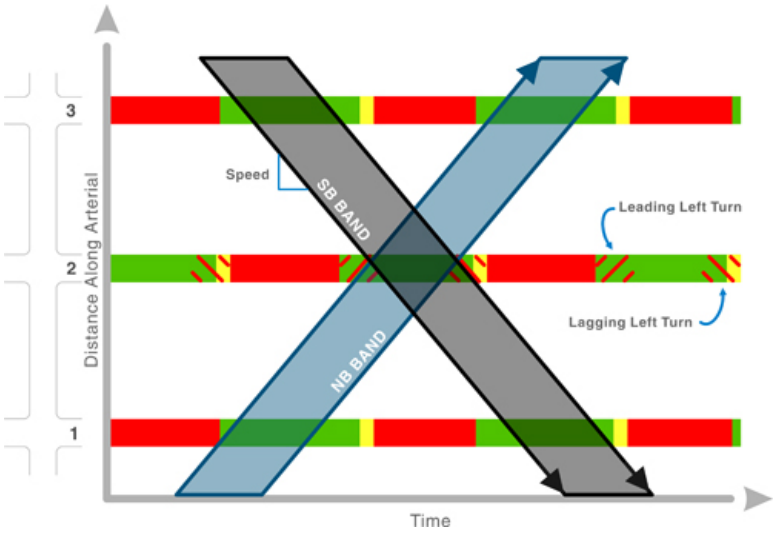

Figure 1.5. Coordination Of Traffic Signals

intersection controllers across an arterial are coordinated to allow continuous traffic flow. Figure 1.5 illustrates the concept of moving vehicles through a series of traffic signals. It is important to note that in the context of traffic management, coordination correspond to the setting of the values for the coordination parameters. It does not correspond to the coordination in multi-agent systems.

1.3 City Of Richardson

The City of Richardson is located fifteen miles north of downtown Dallas and is part of the Dallas-Fort Worth Metroplex. The city has four major highways, eleven major and 6 minor arterial roads, and 128 signalized intersections (see Figure **??**).

1.3.1 Intelight Controllers

The 128 intersection control computers (intersection controllers) are mounted in cabinets at intersections (see Figure 1.7). The intersection controllers are manufactured by Intelight (Intelight, 2019). They run Linux on an ATC-compliant motherboard offering speed, performance and multi-thread capabilities. Intersection controllers provide eight phases to serve

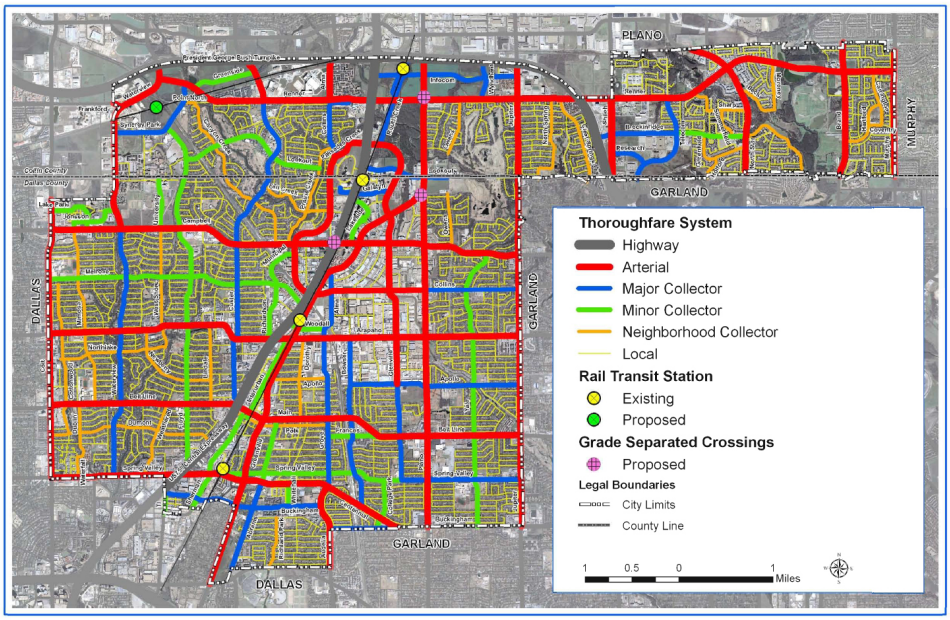

Figure 1.6. City of Richardson's roads map.

standard four-legged intersections (Figure 1.3). Controllert-to-Controller communication links exist but are not used in the current traffic system.

1.3.2 Traffic Management Center

A central traffic management center communicates with the controllers via a WiMAX wireless network operating in the licensed 4.9 GHz public safety band with about 2.5 GB/s total throughput.

1.3.3 Operation Modes

Traffic controllers operate in various modes. During the day, a variety of pre-timed plans designed to address variable traffic patterns are executed based on traffic conditions. Past midnight, controllers operate either in pre-timed, semi-actuated or fully-actuated modes depending on the road types and the existence of a detection system. Minimum greens and maximum greens are specified by traffic engineers for each phase.

Figure 1.7. Intelight Cabinet and Controller (Intelight, 2019).

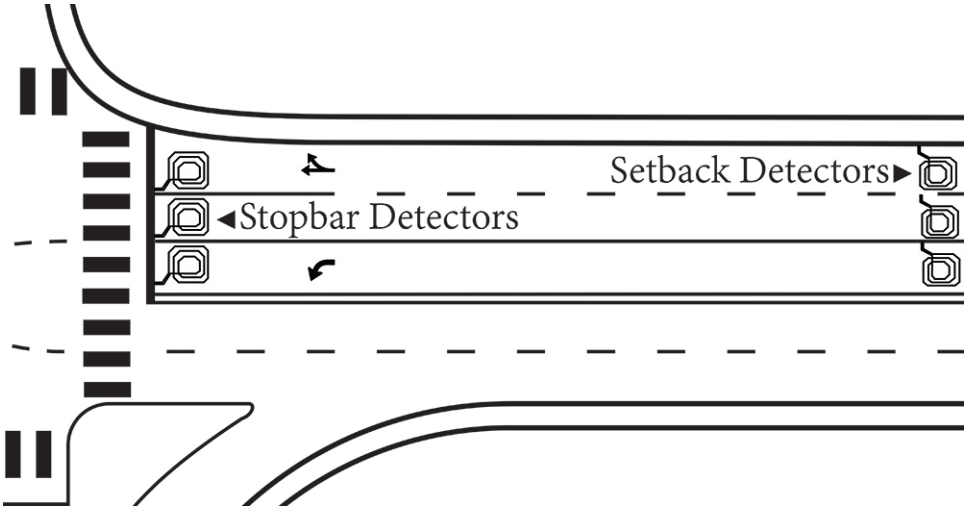

Figure 1.8. Induction Loops in City of Richardson.

1.3.4 Detection System

Vehicles at an intersection are detected through inductive loops. As mentioned in Section 1.2, an inductive loop is a coiled wire that is formed into a loop and installed under the surface of roadways at appropriate distances from the stop bar (see Figure 1.8). When a vehicle passes over the loop or is stopped within its area, a pulse is sent to the traffic signal controller signifying the passage or presence of a vehicle. The City of Richardson utilizes

inductive loops that are either placed right behind the stop bar (i.e., stopbar detectors) or hundreds of feet farther (i.e., setback detectors) As mentioned above, the vehicles that can be realistically detected are those that cross the inductive loop area. Outside the induction loop area, the vehicle positions are not known. In addition, given that detectors are only used at signalized intersections, vehicles traveling to/from residential or service entries are not detected. These limitations impose two significant constraints on the traffic timing system:

1. It is not possible to plan for more than a few seconds. This is due to the fact that observation is limited and it is not possible to accurately predict vehicle arrivals.

2. The system cannot keep a red signal for a long period of time. This is due to the fact that not all lanes are necessarily equipped with stopbar detectors and therefore vehicles waiting for a green signal may go undetected.

1.3.5 Defining Timing Plans

The City of Richardson maintains a traffic count program which conducts scheduled counts on major arterial roads as well as collector streets, i.e., roads which move traffic from local streets to arterial roads. The traffic counts are used for a variety of purposes including the definition of coordinated traffic signal timing along arterial streets. In order to define traffic signal timing plans, traffic engineers assign values to cycle length, offset and splits based on historical data.

1.4 Related Works

The traffic signal timing problem has been traditionally formulated as an optimization problem, i.e., finding the optimal (or near-optimal) values for a set of signal timing parameters with the goal of minimizing an objective function (e.g., vehicle travel time, delay). A plethora of optimization techniques have been discussed in the literature (Papageorgiou et al., 2003).

In addition to the traditional optimal-setting-search-based methods, AI techniques such as game theory (Bazzan, 2005; Cheng et al., 2006), neural networks (Srinivasan et al., 2006; Chao et al., 2008), fuzzy logic (Collotta et al., 2015; Bi et al., 2014), and reinforcement learning (El-Tantawy and Abdulhai, 2010; Abdulhai et al., 2003; Bazzan et al., 2010; El-Tantawy et al., 2013) have been used to propose solutions to the signal timing problem.

In TSTs, signal timing optimizations are computed at the network/sub-network level or at the intersection level.

1.4.1 Optimizations at Network or Sub-Network Level

TSTs in this category include the widely-used TRANSYT (TRANSYT, 2019), SCOOT (SCOOT, 2019) and SCATS (SCATS, 2019), as well as TUC (Diakaki et al., 2002). These systems are centralized, i.e., controllers are managed by a central or several regional computers whose roles are to select the appropriate signal plans. Traffic data are either collected over time and then processed (off-line system), or passed onto a computer in real-time (online system).

TRANSYT is an off-line system which uses historical data to calculate the network's performance index and then applies an optimization process to determine whether changes to the signal settings will improve the index. The main limitation of TRANSYT is the use of historical data which often results in timing plans that are out-of-date and ill-matched to the current traffic conditions.

SCOOT (SCOOT, 2019) is an online, centralized TST used worldwide in more than 200 locations (Zhao and Tian, 2012). Traffic data are collected in real-time through road sensors and passed on to a central computer which predicts queue lengths. The predictions are passed onto an optimizer which determines the optimal timing. Optimizations take effect by incrementally updating a fixed-time plan. Both SCOOT and TRANSYT are responsive control systems with fully centralized control. As such, they are not fit to accommodate highly dynamic traffic patterns and changes in the traffic network.

SCATS (Sydney Coordinated Adaptive Traffic System) (SCATS, 2019) was deployed in Australia in the late 70s. It has been widely used in countries such as the US, China, Singapore and Ireland (Zhao and Tian, 2012). SCATS is structured as a three-layered hierarchical system with a *control center* at the highest level, followed by *regional computers* in the next layer and *local intersection controllers* at the lowest layer. The central computer monitors the system performance whereas the regional computers execute area-based adaptive strategies. Local controllers can modify, within certain limits set by their regional master, their intersection signal settings in response to local traffic conditions. SCATS was primarily designed to respond to time-of-day and long-term variations in traffic. This is achieved by increasing the timings by a few seconds every cycle in response to changes in the traffic conditions. SCATS makes use of real-time measurements from the intersections' incoming roads only. As such SCATS does not perform well when unexpected traffic disruptions occur.

TRANSYT, SCOOT and SCATS were primarily designed to respond to time-of-day and long-term variations in traffic. Their strategy is based on increasing the timings at intersections and does not account for shortening or skipping a phase. In addition, SCOOT and SCATS make use of real-time measurements from the intersections' incoming roads only. As such these systems are not adequate to deal with unexpected traffic disruptions.

TUC (Diakaki et al., 2002) is a recent centralized TST which formulates the traffic control problem as a Linear-Quadratic optimal control problem. TUC considers all traffic intersections simultaneously through the application of a single matrix equation. Results show that TUC is able to achieve highly efficient and extremely simple coordinated control strategies in large traffic networks. Although the system was deployed in the Glasgow area and has proven to be efficient, its centralized architecture requires that the strategy be completely re-designed (i.e., all control matrices be re-calculated) when the traffic network is modified or expanded.

Although partially centralized systems allow intersection controllers to have more decision-making responsibilities, network-level decisions are still made at higher levels.

1.4.2 Optimizations at Intersection Level

TSTs in this category break the signal optimization problem into sub-problems which are assigned to intersection controllers.

Conventional TSTs with No Coordination

Several academic papers have proposed agent-based solutions where isolated smart intersection controllers execute decision making algorithms to benefit their respective intersections (Abdulhai et al., 2003; Mannion et al., 2015a; Chin et al., 2011; Lu et al., 2008; Wen et al., 2007; El-Tantawy and Abdulhai, 2010). The proposed approaches have been validated on simple simulated grid networks or single intersections, using simplistic assumptions about traffic. In addition, optimizations at isolated intersections (without any knowledge about other intersection states) do not guarantee an optimization at the network level.

Conventional TSTs with Implicit Coordination

UTOPIA (Urban Traffic Optimization by Integrated Automation) (Trafitek, 2019) was developed by Mizar Automazione in Turin, Italy and has been used in several countries including Italy, Sweden, Norway and Finland. UTOPIA uses a two-level hierarchical structure. At the lower intersection level, controllers implement signal timings according to the local traffic conditions. The higher *area level* is responsible for setting the network control strategy (i.e., weights for all the elements, minimum and maximum length of each stage, and offsets). The central philosophy of UTOPIA is to provide absolute priority to public transport vehicles and improve the traffic flow for private vehicles, when possible.

PRODYN's (Henry et al., 1983) optimization at the intersection level uses improved forward dynamic programming with constraints on maximum and minimum greens. The coordination between controllers is implicit. It is performed by a) simulating a specific intersection output as soon as the optimization is computed, and sending the simulation

output to each downstream controller; b) using the output message from upstream controllers at the next time step to forecast arrivals. Although PRODYN's approach is conceptually applicable to an entire set of intersection controllers, the exponential complexity of dynamic programming limits its applicability to only a few intersections.

OPAC (Optimized Policies for Adaptive Control) (Gartner et al., 2001) was the first comprehensive strategy to be developed in the U.S. for real-time, adaptive TST. OPAC has gone through several development cycles ranging from OPAC I to OPAC-VFC (Virtual Fixed Cycle). OPAC's intersection controller strategy features a dynamic optimization algorithm that calculates signal timings to minimize a performance function for delay and vehicle stops. The controller's algorithm uses measured as well as predicted traffic data. It determines phase durations that are constrained only by minimum and maximum green times. Similarly to PRODYN, OPAC's earlier versions implement implicit coordination. OPAC suffers from the limitations of dynamic programming.

Conventional TSTs with Explicit Coordination

In OPAC-VFC, the coordination is explicit and is achieved through communication with a central system responsible to identify "critical intersections" and optimizing the cycle length for a group of intersections.

RHODES (Mirchandani and Head, 2001) decomposes the traffic problem into three hierarchical levels. The highest level is the "dynamic network loading" model which captures the slow varying characteristics of traffic (e.g., road closures), and the route selection of travelers. The middle level is the "network flow control" which captures traffic flow characteristics in terms of platoon of vehicles and their speed. The lower level is the "intersection control" which captures fast varying traffic characteristics in terms of individual vehicles. Each level makes use of prediction models.

RHODES and OPAC do not employ defined traffic cycles or signal timing plans. They utilize traffic flow models that predict vehicle arrivals at the intersection, and adjust the

timing of each phase to optimize an objective function. Because they emphasize traffic prediction, these systems can respond to the natural statistical variations in traffic flow as well as to flow variations caused by traffic incidents or other unpredictable events. Intersection control equipment for these systems is more complex and not readily available in the field.

AI-Based TSTs

With respect to intersection-level TSTs that implement AI-based techniques, most recent approaches are research-oriented and heavily based on the use of the multi-agent system paradigm. The core concept for these systems is that intersection controllers are controlled by autonomous software agents that are capable of interacting with one another to achieve a local or global goal. Models and architectures have been presented (France and Ghorbani, 2003; Roozemond, 2001; Mashayekhi and List, 2015), and solutions with various techniques have been discussed in the literature (e.g., game theory (Bazzan, 2005; Bui et al., 2017; Elhenawy et al., 2015; Cheng et al., 2006; Chen and Ben-Akiva, 1998; Zohdy and Rakha, 2012; Zhen-long, 2003; De Oliveira et al., 2005; Sun et al., 2006), neural networks (Ghanim and Abu-Lebdeh, 2015; Chao et al., 2008; Srinivasan et al., 2006; Saito and Fan, 2000), fuzzy logic (Collotta et al., 2015; Bi et al., 2014; Kosonen, 2003), reinforcement learning (Li et al., 2016; Bazzan et al., 2010; El-Tantawy et al., 2013; Mannion et al., 2015b; Teodorović et al., 2006; Dong et al., 2005; Abdulhai et al., 2003)). Unfortunately, these solutions are based on assumptions that simplify the traffic signal optimization problem and were validated on simple networks. Only a very few multi-agent solutions have considered the full spectrum of real-world traffic constraints and were validated on simulated models of real cities. Emerging RL approaches have been shown to be well-fitted to TST systems (Bazzan, 2009). Therefore, we restrict our discussion to the systems in which agents are placed in intersection controllers and implement RL approach. Detailed discussions of agent-based TSTs can be found in (Chen and Cheng, 2010; Bazzan and Klügl, 2014; Araghi et al., 2015; Liu, 2007; Li et al., 2014; Mannion et al., 2016; Yau et al., 2017; Chin et al., 2011).

In (Bazzan et al., 2010) Bazzan et al. used a multi-agent reinforcement learning approach. The multi-agent reinforcement learning problem considers not individual states and actions, but joint states and actions, for any number of agents. The general approach naturally leads to an exponential number of $< state, action >$ pairs. In order to address this problem, Bazzan et al. propose to partition the set of intersection controller agents into groups of three and assign a supervisor agent to the group to determine possible joint actions. For the sake of preventing a combinatorial explosion in the number of $< state, action >$ pairs, each intersection can only be in one of three "coarse-grained" states. Every sixty seconds, each intersection controller determines its $< state, action >$ pair. The supervisor who observes the agents retrieves a set of actions that yielded maximum rewards in the past and communicates it to the agents. Experiments using a grid-structured road network composed of 64 nodes (i.e., intersections) connected through unidirectional links (i.e., one way road) were run. Even though the experiments show an improvement over the author's single-agent approach (Bazzan et al., 2010), the assumptions make this work unfit for real-world traffic systems.

The basic premises of RL-based approaches is that traffic signal timing is not pre-defined but agents learn the appropriate traffic signal settings. In (Dusparic and Cahill, 2012), Dusparic and Cahill discuss DWL, a multi-agent RL-based algorithm for multi-policy optimization. In DWL, agents collaborate to satisfy multiple heterogeneous policies simultaneously (e.g., prioritize buses, reduce pollution). An agent uses a combination of Q-learning and W-learning processes for each of its local policies, and to learn the suitability of its actions for each of its direct neighbor's policies. Collaboration is implicit and restricted to immediate neighbors. It is achieved through the concept of "remote policy". With respect to traffic signal timing, the authors state that the optimization is related to phase selection but no detail is provided about the process. The emphasis of this paper is more on the RL-based multi-policy optimization than signal timing optimization. DWL was evaluated on a simulated map of Dublin's inner city including 62 signalized intersections. The policies used

in the evaluation are a policy that optimizes global waiting time and a policy that priotitizes public transport vehicles. Experiments that were run using artificially generated data show that DWL outperforms the traditional fixed-timed approach and the Simple Adaptive Technique (Richter, 2006).

In (Dusparic et al., 2016), the authors extend DWL to consider the optimization of phase duration. This optimization is only possible if more fine-grained traffic data (i.e., precise traffic counts) are available. The extended DWL, called REALT, was evaluated in VISSIM (PTV-Group, 2018) on a simulated model of Cork City comprising six intersections. The same policies as in (Dusparic and Cahill, 2012) were implemented. Experiments that were run using real-world data show that REALT outperforms SCOOT in terms of delay and number of stops.

El-Tantawy et al. (El-Tantawy et al., 2013) present a coordinated multi-agent reinforcement learning architecture called MARLIN-ATSC. In MARLIN-ATSC, agents can operate in either independent or integrated mode. Coordination is implicit and achieved through multi-agent modular Q-learning. In modular Q-learning, the state space is partitioned into partial state spaces comprising of two agents. An agent learns a joint policy with only one of it direct neighbors. With respect to traffic signal timing, the optimization is related to the selection of a phase among a set of pre-defined phases. MARLIN-ATSC was tested on a simulated network of the Lower Downtown Toronto network comprising 59 intersections. Real-world data for about 25000 vehicle trips during morning peak hours were used to evaluate the system. Experimental results show that MARLIN-ATSC reduces the average intersection delay compared to a basic signal timing used by the City of Toronto. The main drawback of MARLIN-ATSC is the assumption that an intersection controller can only consider the interest of one immediate neighboring controller.

In addition to the limitations discussed above, both (El-Tantawy et al., 2013) and (Dusparic et al., 2016) assume that the systems may have variable phasing sequence. While this

assumption may be reasonable in a simulated environment, it is not acceptable in a real-world setting (Koonce et al., 2008). A variable phasing sequence can lead to endless greens for phases with continuous demand.

This dissertation advances the state-of-the-art as follows:

1. It discusses DALI, a collaborative multi-agent traffic signal system where: i) collaboration between agents is explicit (i.e., through communication and negotiation); ii) the collaboration scope is not limited to direct neighbors and is defined dynamically based on traffic conditions.

2. Agents communicate with one another through direct links and do not have a supervising component to oversee coordination.

3. DALI was thoroughly validated through simulation on a realistic model of a city's traffic network comprising of 1365 road segments and 128 intersections. It was also validated through hybrid simulation.

4. DALI's model follows a"plug-and-play" approach where agents are plugged into existing intersection controllers and can be turned on and off.

5. DALI was deployed and field-tested. The deployment considered safety constraints.

6. We share the lessons learned in developing and deploying a collaborative multi-agent traffic signal system.

To our knowledge, DALI is the first collaborative multi-agent based system to be field-tested in the US.

CHAPTER 2

DALI'S ALGORITHMS[1]

In this chapter, we start by giving an overview of DALI's model and then present the detailed algorithms of DALI's agents.

2.1 DALI Model

In DALI, agents communicate with each other through direct links and do not have a supervising agent to oversee coordination. Agents have knowledge of the traffic network topology. They receive information about the incoming traffic flow from their neighboring controllers and determine the outgoing traffic flow based on the data sensed by their inductive loops (see Figure 2.1). Intersections are assigned weights to indicate their criticality in the traffic network.

Agents continuously exchange the information of detected vehicles and use it to compute all possible timing plans for the near future (i.e. observable horizon). For an intersection c_n, observable horizon is defined as the duration of travel to c_n, from its farthest neighbor with the maximum allowed speed. A plan include a sequence of phase combinations and the start and the end of their green interval. The agent considers timing configuration (e.g. agreed split, minimum and maximum green, yellow and all red intervals) in the computation of the possible plans. Then, agents assign a score to each plan based on the queue length of its road lanes and the estimated vehicle arrivals. They finally execute the best plan and start the process over immediately.

At any given time, if an agent determines that its intersection is congested, it deliberates and defines a timing configuration to alleviate congestion by adjusting splits. Then, it

[1] ©2018 IEEE. Portions Adapted, with permission, from Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. "A Self-Adaptive Collaborative Multi-Agent based Traffic Signal Timing System." In Proceedings of the 4th IEEE International Smart Cities Conference, ISC2 2018, September 2018.

broadcasts the configuration to the neighboring agents. Upon receipt, the agents evaluate the configuration by calculating its effect on each outgoing road of their intersection. They communicate the information with the affected neighboring agents. And the process iterates until it either reaches a) an intersection within the city boundaries for which the effect of the request is below a threshold or b) an exit junction at the city's boundaries. The information is then propagated back, and at each stage of the propagation, the agents consider each other's feedback for their decision on their *level of agreement* with the configuration, i.e., a value which indicates the extent at which an agent can agree with the terms of the configuration. The initiating agent then decides whether to execute or ignore the configuration. It proceeds by informing the agents of its final decision and, in case the configuration is to be applied., requests that they update their timing.

2.2 Model Definitions

2.2.1 Set Definitions

- $T = \{t_1, .., t_i\}$ is the set of time-stamps at which traffic conditions are evaluated.
- $C = \{c_1, .., c_n\}$ is the set of intersection controllers. An intersection controller c_n is assigned a weight ω which corresponds to its priority in the road network.
- $Rd = \{r_{c_1,c_2}, .., r_{c_m,c_n}\}$ is the set of road segments between intersections. A road segment r_{c_m,c_n} is defined in terms attributes such as length l, speed limit sl and a set of lanes $LN_{r_{c_m,c_n}} = \{ln_1..ln_q\}$.
- $LT_{r_{c_m,c_n}.ln_w}$ is the set of lanes that are accessible from $r_{c_m,c_n}.ln_w$.
- $LF_{r_{c_m,c_n}.ln_w}$ is the set of lanes that have access to $r_{c_m,c_n}.ln_w$.
- Each lane $r_{c_m,c_n}.ln_w$ has a detector $r_{c_m,c_n}.ln_w.d$. The detector $r_{c_m,c_n}.ln_w.d$ could be either setback bk or stopbar tr.

- $PH_{c_n} = \{ph_{c_n,1}, ..ph_{c_n,k}\}$ is the set of phases for the intersection controlled by c_n. A phase $ph_{c_n,k}$ is defined in terms of γ, the split time, ν, the minimum green time, η, the maximum green time, ϵ, the yellow time, ξ, the red time and $LN_{ph_{c_n,k}}$, the set of lanes it applies to.

- A *PlanItem* is defined as a combination of non-conflicting phases, the start (t_{start}) and the end (t_{end}) of the time that they get green, and *duration*, the duration of green.

- A *Plan* is a sequence of *PlanItem*.

- S_{Plan} is a real value that represents a plan's score.

- $VL_{r_{c_m,c_n}} = v_1, .., v_k$ is the set of vehicles that will arrive at intersection n from r_{c_m,c_n}.

- $EV_{ENT,EVT,Ind}$ indicates an event at the intersection. An event EV happens for entity ENT *phase* or *detector*. For a phase, the event type EVT is *green*, *yellow* or *red*. For a detector, EVT is *active* or *inactive*. Ind specifies the index of the occurrence of the same event in the past. For example, $EV_{ph_{c_n,k},green,2}$ refers to the past second time that phase $ph_{c_n,k}$ was $green$.

- *Gap* is a constant that represents the maximum time gap between consecutive vehicles.

2.2.2 Function Definitions

- $p(r_{c_m,c_n}.ln_w, r_{c_n,c_p})$ is the probability that a vehicle exiting lane w in road segment r_{c_m,c_n} enters the road segment r_{c_n,c_p}. This probability is computed by traffic engineers based on historical data.

- $p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)$ is the probability that a vehicle which enters road segment r_{c_m,c_n}, leaves it from lane w. This probability is also computed by traffic engineers based on historical data.

- $t(EV_{ENT,EVT,Ind})$ is the time that event $EV_{ENT,EVT,Ind}$ has happened.

- $q(ph_{c_n,k}, t)$ is the queue length of phase $ph_{c_n,k}$ at time t. It is computed as the sum of queue lengths of the lanes that $ph_{c_n,k}$ controls.

- $passed(EV_{ph_{c_n,k},red,1}, r_{c_m,c_n}.ln_w.d)$ is the number of vehicles that passed over detector $r_{c_m,c_n}.ln_w.d$ while its phase was red.

- $passedNS(EV_{ph_{c_n,k},green,1}, r_{c_m,c_n}.ln_w.d)$ is the number of vehicles that have continuously passed detector $r_{c_m,c_n}.ln_w.d$ since the phase got green.

- $rateOut(r_{c_m,c_n}.ln_w)$ is the rate of vehicles (per second) that leave the intersection through lane w of road segment r_{c_m,c_n}.

- $rateIn(t_i, r_{c_m,c_n})$ is the rate of vehicles (per second) that enter road segment r_{c_m,c_n} in the evaluation interval τ that ends at time t_i.

- $\xi_{t_i,r_{c_m,c_n}.ln_w}$ is the *traffic throughput* for lane $r_{c_m,c_n}.ln_w$, i.e., the ratio of vehicles getting in and leaving the lane. It is defined as:

$$\xi_{t_i,r_{c_m,c_n}.ln_w} = \frac{rateIn(t_i, r_{c_m,c_n})}{rateOut(t_i, r_{c_m,c_n}.ln_w)} \times p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)$$

2.3 Non Congested Traffic Conditions

On a continuous basis, agent c_n uses the information received, the observations and the timing configurations (e.g., minimum and maximum green, yellow and all red intervals) to compute possible timing plans for the observable horizon. A timing plan *Plan* includes a sequence of phase combinations as well as the start and end of their green intervals, and the green duration. Once the timing plans are computed, agent c_n assigns a score to each plan based on its phases' estimated vehicle arrivals and the phases' estimated queue lengths. Then, c_n executes the plan with the lowest score and re-starts the process immediately.

2.3.1 Communicating Vehicle Arrivals

With respect to *vehicle arrivals*, c_n continuously exchanges information about detected vehicles with its neighbors (see Algorithm 1). For instance, when phase $ph_{c_n,k}$ that controls $r_{c_m,c_n}.ln_w$ is green, c_n sends a message about vehicle arrival to the neighboring agents that are accessible from $r_{c_m,c_n}.ln_w$ under the following conditions: 1) c_n has not sent the information of a detected vehicle on this lane in the last *Gap* seconds. 2) The queue length of $r_{c_m,c_n}.ln_w$ is greater than zero or detector $r_{c_m,c_n}.ln_w.d$ gets deactivated. For each neighbor c_p the message contains the detection time and $p(r_{c_m,c_n}.ln_w, r_{c_n,c_p})$. Agent c_n estimates the arrival of vehicles for its phases based on the information that it receives from its own neighbors.

Algorithm 1: Communicate Detected Vehicles

Require: PH_{c_n}, t

for all $ph_{c_n,k} \in PH_{c_n}$ **do**
  for all $r_{c_p,c_n}.ln_w \in LN_{ph_{c_n,k}}$ **do**
    if $HasNotSentInTheLastGapSeconds$ **then**
      if $ph_{c_n,k} \neq red$ **then**
        if $q(r_{c_m,c_n}.ln_w, t) > 0$ **then**
          $Send(c_m, t, p(r_{c_p,c_n}.ln_w, r_{c_n,c_m}))$
        end if
      else
        if $t(EV_{r_{c_p,c_n}.ln_w.d,inactive,1}) < GAP)$ **then**
          $Send(c_m, t, p(r_{c_p,c_n}.ln_w, r_{c_n,c_m}))$
        end if
      end if
    end if
  end for
end for

2.3.2 Queue Length Estimation

With respect to *queue length*, c_n estimates the queue length of lane $r_{c_m,c_n}.ln_w$ based on its detector type (i.e., *stopbar* or *setback*) and the status of its phase $ph_{c_n,k}$ (i.e., green, red, yellow). In case the detector is of type *stopbar* and $ph_{c_n,k}$ is green, then the queue length of

lane $r_{c_m,c_n}.ln_w$ at time t is estimated as:

$$q(r_{c_m,c_n}.ln_w, t) = q(r_{c_m,c_n}.ln_w, t(EV_{ph_{c_n,k},g,1}))$$
$$-passedNS(EV_{ph_{c_n,k},green,1}, r_{c_m,c_n}.ln_w.d)$$

If $ph_{c_n,k}$ is red, the queue is computed as:

$$q(r_{c_m,c_n}.ln_w, t) = rateOut(r_{c_m,c_n}.ln_w)$$
$$*(t - t(EV_{ph_{c_n,k}}, r, 1))) + q(r_{c_m,c_n}.ln_w, t(EV_{ph_{c_n,k},r,1}))$$

In case the detector is of type *setback* and phase $ph_{c_n,k}$ is green, then the queue length of $r_{c_m,c_n}.ln_w$ is estimated as:

$$q(r_{c_m,c_n}.ln_w, t) = q(r_{c_m,c_n}.ln_w, t(EV_{ph_{c_n,k},g,1}))$$
$$-(GAP \times (t - t(EV_{ph_{c_n,k},g,1})))$$

When $ph_{c_n,k}$ is red, the queue is computed as:

$$q(r_{c_m,c_n}.ln_w, t) = passed(EV_{ph_{c_n,k},red,1}, r_{c_m,c_n}.ln_w.d)$$
$$+q(r_{c_m,c_n}.ln_w, t(EV_{ph_{c_n,k},r,1}))$$

2.3.3 Evaluating Timing Plans

Agents continuously generate all possible timing plans concerning its timing configuration. It then uses the algorithm 2 to evaluate possible plans. Agent c_n assigns a score to each plan. This score is computed as follows: For each $PlanItem$ in $Plan$, for each phase $ph_{c_n,k}$ that is not in $PlanItem$:

$$S_{Plan} \leftarrow S_{Plan} + q(ph_{c_n,k}, t) \times PlanItem.duration$$

Also, for each road lane $r_{c_m,c_n}.ln_w$ in $LN_{ph_{c_n,k}}$, and vehicle arrival v_k in $VL_{r_{c_m,c_n}}$, if $v_k.arrival$ is between $PlanItem.t_{start}$ and $PlanItem.t_{end}$:

$$S_{Plan} \leftarrow S_{Plan} + (PlanItem.t_{end} - v_k.arrival)$$
$$\times p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)$$

Finally c_n executes the plan with the lowest score and re-starts the process immediately.

Algorithm 2: Evaluate Plans

Require: $Plan, t$

1: $S_{Plan} \leftarrow 0$
2: **for all** $ph_{c_n,k} \in PH_{c_n}$ **do**
3: $q_{ph_{c_n,k}} \leftarrow q(ph_{c_n,k}, t)$
4: $S_{ph_{c_n,k}} \leftarrow 0$
5: **for all** $PlanItemPI \in Plan$ **do**
6: **if** $ph_{c_n,k} \in PI$ **then**
7: $q_{ph_{c_n,k}} \leftarrow max(q_{ph_{c_n,k}} - GAP \times PI.durtion, 0)$ **else**
$S_{ph_{c_n,k}} \leftarrow S_{ph_{c_n,k}} + (q_{ph_{c_n,k}} \times PI.durtion)$
9: **for all** $r_{c_m,c_n}.ln_w$ *in* $LN_{ph_{c_n,k}}$ **do**
10: **for all** v_k *in* $VL_{r_{c_m,c_n}}$ **do**
11: **if** $v_k.arrival < PI.t_{end}$ **then**
12: **if** $v_k.arrival > PI.t_{start}$ **then**
13: $del \leftarrow (PI.t_{end} - v_k.arrival)$
14: $del \leftarrow del \times p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)$
15: $S_{ph_{c_n,k}} \leftarrow S_{ph_{c_n,k}} + del$
16: $S_{ph_{c_n,k}} \leftarrow S_{ph_{c_n,k}} + SC$
17: **end if**
18: **end if**
19: **end for**
20: **end for**
21: **end if**
22: **end for**
23: $S_{Plan} \leftarrow S_{Plan} + S_{ph_{c_n,k}}$
24: **end for**

2.4 Congested Traffic Conditions

In DALI, agents collaborate with one another to dynamically respond to traffic changes. In this section, we discuss the agent algorithms at the basis of the collaborative approach. The algorithms make use of thresholds a, b, h, d, e, f and g which are assigned values based on historical traffic data.

Algorithm 3: Controller Congestion Reduction

Require: PH_{c_n}, t_i

1: **for** $ph_{c_n,k} \in PH_{c_n}$ **do**
2: $EvaluateTraffic(ph_{c_n,k}, t_i : TotalInstCong)$
3: **if** $\frac{TotalInstCong}{b} > d$ **then**
4: $GeneratePlan(ph_{c_n,k}, t_i : conf_{new})$
5: $RequestForEvaluation(ph_{c_n,k}, conf_{new} : \Psi_{c_n})$
6: **if** $\Psi_{c_n} > h$ **then**
7: $ExecutePlan(conf_{new})$
8: **end if**
9: **end if**
10: **end for**
11: **if** $ReceiveRequestForEvaluation(c_p, \kappa_{r_{c_p,c_n}}, \kappa_{ph_{c_q,j}})$ **then**
12: $ComputeLevelOfAgreement(\kappa_{r_{c_p,c_n}}, \kappa_{ph_{c_q,j}})$
13: **end if**
14: **if** $ReceiveRequestForExecution(c_p, conf_{new})$ **then**
15: $AdjustTiming(conf_{new})$
16: **end if**

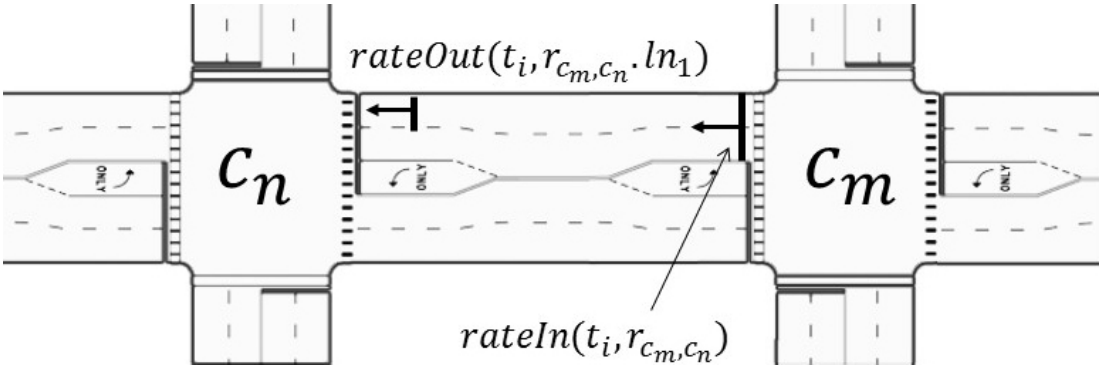

Figure 2.1. c_n determines *rateOut* and receives *rateIn* from c_m.

2.4.1 Detecting Congestion

Intersection controller c_n continuously evaluates the traffic state by executing Algorithm 3 to determine if a re-timing operation is necessary. As shown in Figure 2.1, at each t_i, c_n receives $rateIn$ (determined by its neighbors' detectors) and determines $rateOut$. At time t_i,

Algorithm 4: Evaluate Traffic

Require: $ph_{c_n,k}, t_i$

1: $TotalInstCong \leftarrow 0$
2: **for** $j = 0$ *to* b **do**
3: $\quad \delta = 0$
4: $\quad$ **for** $r_{c_m,c_n}.ln_w \in LN_{ph_{c_n,k}}$ **do**
5: $\quad\quad \delta \leftarrow \xi_{ti-j,r_{c_m,c_n}.ln_w} + \delta$
6: $\quad$ **end for**
7: $\quad Cong_{t_i,ph_{c_n,k}} \leftarrow \delta$
8: $\quad$ **if** $Cong_{t_i,ph_{c_n,k}} \geq a$ **then**
9: $\quad\quad TotalInstCong \leftarrow TotalInstCong + 1$
10: $\quad\quad$ * TotalInstCong Represent Sum Over InstantCongestion
11: $\quad$ **end if**
12: **end for**

controller c_n computes $Cong_{t_i,ph_{c_n,k}}$ as the average throughput for the set of lanes controlled by $ph_{c_n,k}$ (see Algorithm 4).

$$Cong_{t_i,ph_{c_n,k}} = \sum_{r_{c_m,c_n}.ln_w \in LN_{ph_{c_n,k}}} \xi_{t_i,r_{c_m,c_n}.ln_w}$$

If $Cong_{t_i,ph_{c_n,k}}$ is greater than threshold a, then c_n considers that there is an *instant congestion* and assigns the value of 1 to $InstantCongestion$ defined as:

$$InstantCongestion_{t_i,ph_{c_n,k}} = \begin{cases} 1 & Cong_{t_i,ph_{c_n,k}} \geq a \\ 0 & Cong_{t_i,ph_{c_n,k}} < a \end{cases}$$

It proceeds by considering the past b evaluation cycles to determine the percentage of evaluation cycles in which the phase was congested (see Figure 2.2). This is defined as:

$$PercentCong_{t_i,ph_{c_n,k}} = \frac{\sum_{j=i-b}^{i} InstantCongestion_{t_j,ph_{c_n,k}}}{b} \times 100$$

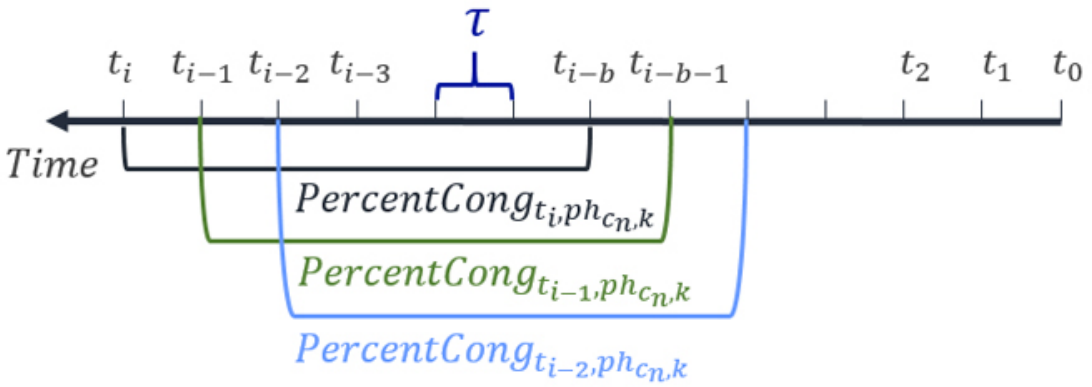

Figure 2.2. *PercentCong* of phase $ph_{c_n,k}$.

If $PercentCong_{t_i,ph_{c_n,k}} > d$ then the road lanes controlled by $ph_{c_n,k}$ are considered to be congested.

2.4.2 Defining new configuration

The controller deliberates to determine the value of a new split that will alleviate congestion on $ph_{c_n,k}$. This is achieved in step 7 of Algorithm 5. The value of the new split is calculated by agent as:

$$conf_{new}.phase.\gamma = conf_{cur}.phase.\gamma \times (e + \frac{\sum_{j=i-\nu}^{i} Cong_{t_j,ph_{c_n,k}}}{\nu} \times f)$$

e and f are coefficients that can be calibrated. They regulate the influence of the traffic throughput and the current split time for the new split time. Values of cycle length and offset change with respect to the new split. If $conf_{new}.phase.\gamma$ is greater than the maximum allowed split time γ_{MAX} defined for phase $ph_{c_n,k}$ as:

$$ph_{c_n,k}.\gamma_{MAX} = ph_{c_n,k}.\eta + ph_{c_n,k}.\epsilon + ph_{c_n,k}.\xi$$

then its value is set to $ph_{c_n,k}.\gamma_{MAX}$ (step 9).

Algorithm 5: Compute New Configuration

Require: $ph_{c_n,k}, t_i$
Ensure: $conf_{new}$
1: $conf_{new}.phase \leftarrow ph_{c_n,k}$
2: $\chi \leftarrow 0$
3: **for** $j = i - \nu$ *to* i **do**
4: $\quad \chi \leftarrow \chi + Cong_{t_j,ph_{c_n,k}}$
5: **end for**
6: $\chi \leftarrow \frac{\chi}{\nu}$
7: $conf_{new}.phase.\gamma \leftarrow conf_{cur}.phase.\gamma * (e + \chi * f)$
8: **if** $conf_{new}.phase.\gamma > ph_{c_n,k}.\gamma_{MAX}$ **then**
9: $\quad conf_{new}.phase.\gamma \leftarrow ph_{c_n,k}.\gamma_{MAX}$
10: **end if**

2.4.3 Requesting Agents' Feedback

c_n determines the impact of executing the new configuration on the neighboring intersections in terms of κ, the increment in vehicle rate. $\kappa_{r_{c_m,c_n}.ln_w}$ is calculated for road lane $r_{c_m,c_n}.ln_w$ as:

$$\kappa_{r_{c_m,c_n}.ln_w} = \frac{rateOut(t_i, r_{c_m,c_n}.ln_w)}{conf_{new}.phase.\gamma} \times (conf_{new}.phase.\gamma - conf_{cur}.phase.\gamma)$$

$\kappa_{ph_{c_n,k}}$ for a phase $ph_{c_n,k}$ is defined as the sum of $\kappa_{r_{c_m,c_n}.ln_w}$ for the set of lanes controlled by the phase (Algorithm 6, Step 3). In the same way, $\kappa_{r_{c_n,c_p}}$ for a road segment r_{c_n,c_p}, is the sum of $\kappa_{r_{c_n,c_p}.ln_w}$ (Algorithm 6, Step 10).

Controller c_n proceeds by sending $conf_{new}$, $\kappa_{r_{c_n,c_p}}$ and $\kappa_{ph_{c_n,k}}$ to each adjacent controller c_p for evaluation. $\kappa_{ph_{c_n,k}}$ corresponds to the increment in the rate of vehicles that exit the road lanes controlled by $ph_{c_n,k}$, in case the new configuration is to be executed. $\kappa_{r_{c_n,c_p}}$ corresponds to the portion of $\kappa_{ph_{c_n,k}}$ that goes to road segment r_{c_n,c_p}.

2.4.4 Computing Level Of Agreement

Upon receipt of a new configuration, c_n's neighboring controller c_p computes $\kappa_{r_{c_p,c_q}}$ for each of its neighbor controllers c_q and requests that they each evaluate the configuration. The process

Algorithm 6: Request for Evaluation

Require: $ph_{c_n,k}, conf_{new}$
Ensure: Ψ_{c_n}
1: $\kappa_{ph_{c_n,k}} \leftarrow 0$
2: **for** $r_{c_m,c_n}.ln_w$ *in* $LN_{ph_{c_n,k}}$ **do**
3: $\quad \kappa_{ph_{c_n,k}} \leftarrow \kappa_{ph_{c_n,k}} + \kappa_{r_{c_m,c_n}.ln_w}$
4: **end for**
5: $\Psi_{c_n} \leftarrow 0$
6: **for** *accessible neighbor* c_p *, in parallel* **do**
7: $\quad \kappa_{r_{c_n,c_p}} \leftarrow 0$
8: $\quad$ **for** $r_{c_m,c_n}.ln_w \in LN_{ph_{c_n,k}}$ **do**
9: $\quad\quad$ **for** $r_{c_n,c_p}.ln_u \in LT_{r_{c_m,c_n}.ln_w}$ **do**
10: $\quad\quad\quad \kappa_{r_{c_n,c_p}} \leftarrow \kappa_{r_{c_n,c_p}} + (p(r_{c_m,c_n}.ln_w, r_{c_n,c_p}.ln_u) \times \kappa_{r_{c_m,c_n}.ln_w})$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: $\quad$ **Send**(c_p, $\kappa_{r_{c_n,c_p}}$, $\kappa_{ph_{c_n,k}}$)
14: $\quad$ **Receive**(c_p, Ψ_{c_p})
15: $\quad \Psi_{c_n} \leftarrow \Psi_{c_n} + \Psi_{c_p}$
16: **end for**

propagates until at a given intersection, either the value of κ is smaller than threshold g or the configuration reaches the road network boundaries. Following this step and recursively, each controller sends back its level of agreement in terms of a real number Ψ, to the controller from which it has received the request. An intermediate controller, c_p, calculates Ψ_{c_p} based on the existing traffic throughput, its priority ω and the ratio of the received additional vehicle throughput (see Algorithm 7). x, y and z are coefficients that calibrate the influence of variables in Ψ. After receiving the level of agreement from all affected neighbors, c_p adds them to its own level of agreement Ψ_{c_p} and sends the value back to c_n. The final decision is made based on the value of Ψ_{c_n} representing the opinion of all affected controllers in the network.

Algorithm 7: Compute Level Of Agreement

Require: $\kappa_{r_{c_n,c_p}}, \kappa_{ph_{c_n,k}}$
Ensure: Ψ_{c_p}
1: $\Psi_{c_p} \leftarrow 0$
2: **for** $r_{c_n,c_p}.ln_u \in LN_{r_{c_n,c_p}}$ **do**
3: $\quad \Psi_{c_p} \leftarrow \Psi_{c_p} + x \times \omega(c_p) \times \frac{\kappa_{r_{c_n,c_p}}}{\kappa_{ph_{c_n,k}}} \times (y - z \times \frac{(\kappa_{r_{c_n,c_p}} + rateIn(t_i, r_{c_n,c_p}) \times p(r_{c_n,c_p}, r_{c_n,c_p}.ln_u)}{rateOut(t_i, r_{c_n,c_p}.ln_u)})$
4: **end for**
5: **for** *accessible neighbor* c_q *from* c_p *, in parallel* **do**
6: $\quad \kappa_{r_{c_p,c_q}} \leftarrow 0$
7: $\quad$ **for** $r_{c_n,c_p}.ln_u \in LN_{r_{c_n,c_p}}$ **do**
8: $\quad\quad$ **for** $r_{c_p,c_q}.ln_f \in LF_{r_{c_n,c_p}.ln_u}$ **do**
9: $\quad\quad\quad \kappa_{r_{c_n,c_p}} \leftarrow \kappa_{r_{c_n,c_p}} + p(r_{c_n,c_p}, r_{c_n,c_p}.ln_u) \times p(r_{c_n,c_p}.ln_u, r_{c_p,c_q}.ln_f) \times \kappa_{r_{c_n,c_p}}$
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: $\quad$ **if** $\kappa_{r_{c_n,c_p}} > g$ **then**
13: $\quad\quad$ **Send**$(c_q, \kappa_{r_{c_n,c_p}}, \kappa_{ph_{c_n,k}})$
14: $\quad\quad$ **Receive**(c_q, Ψ_{c_q})
15: $\quad\quad \Psi_{c_p} \leftarrow \Psi_{c_p} + \Psi_{c_q}$
16: $\quad$ **end if**
17: **end for**
18: **Send**(c_n, Ψ_{c_p})

2.4.5 Special Cases

During the execution of the scenario discussed above, several exceptions may occur. These include the following:

1. A controller may receive more than one configuration to evaluate at the same time. In this case, the controller evaluates the configuration sent by the controller with the highest priority and halts the evaluation of other configurations. If the configurations were sent by neighbor controllers having the same level of criticality, the controller selects one according to a pre-defined criteria, e.g., the smaller controller ID.

2. A controller may receive more than one request to evaluate the same configuration. In the example mentioned above, executing c_2's configuration will increase the throughput

of both r_{c_2,c_3} and r_{c_5,c_3}. This will result in c_3 receiving an evaluation request first from c_2 and then from c_5. Controller c_3 evaluates the request from c_2 and stores the received additional throughput value. Then c_3 considers the stored value to evaluate the request of c_5.

3. A controller may lose connection from the network and stop responding to requests of evaluation. In this case, other agents assume that the disconnected agent fully disagrees with any retiming configuration.

4. When a configuration gets rejected, the main agent generates a new configuration by reducing the split of the rejected configuration and asking other agents to evaluate the new configuration.

5. After executing a new timing configuration, when the traffic situation goes back to normal, the main agent switches back to the original timing configuration and asks other agents to do the same.

2.5 Adaptive Assignment of Threshold Values

2.5.1 Overview of Reinforcement Learning

Reinforcement Learning (RL) allows software agents to learn using temporal learning without the need for external supervision. Several approaches for temporal learning have been proposed. Most traffic signal controllers discussed in the literature use Q-learning.
At time t, an agent observes its environment modeled as a Markov Decision Process. Based on the observed environment state $s \in S$, it takes an action $a \in A$ and then receives a delayed or discounted reward at time $t + 1$.

The Q-value of a <state,action> pair represents the appropriateness of taking action a in state s in the long term. It is maintained in a Q-table with $|S| \times |A|$ entries using the Q-function.

During action selection, an agent selects the best known action that maximizes value function $V(s) = \max_{a \in A} Q(s, a)$. After the execution of an action, the agent uses the Q-function to update its Q-value based on the maximum Q-value of the next state s'. The Q function is commonly defined as:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r(s') + \gamma \max_{a \in A} Q(s', a))$$

where α is the learning rate that controls how fast the reward values are modified, $0 < \gamma < 1$ is the discount factor that controls how far the algorithm looks into the future, and $\gamma \max_a Q(s', a)$ is the highest possible accumulated reward expected to be received.

At the start of the system, an agent initializes the Q-values to a fixed value. Then, it continuously selects the action with the maximum reward in the current state and updates the previous estimates of Q-values.

2.5.2 Agent Algorithms for Adaptive Threshold Assignment

The algorithms discussed in Section 2.4 assign fixed values to various thresholds (i.e., a, b, h, d, e, f and g) based on historical data. In order to make the collaboration process between agents more adaptive, it is necessary that the agents adjust the values dynamically. In this section we discuss how an RL-based approach is used to dynamically select favorable values for thresholds a, d and g. As mentioned in Section 2.4, threshold a controls the agent's sensitivity to detecting congestion. The lower the values of a, the higher the likelihood for an agent to detect congestion. Threshold d controls the time duration that a phase should be flagged as congested in order to be considered as congested. Finally, g controls the collaboration scope. With lower values of g, a higher number of agents will be involved in the decision-making for a new timing plan.

Environment State

Given that queue lengths are not measurable in a real-world TST, we use the traffic flow rate to define the environment state. As mentioned in Section 2.4, at time t_i, the traffic flow rate for road segment r_{c_m,c_n} is defined as:

$$\xi_{r_{c_m,c_n}.ln_w}(t_i,\tau) = \frac{rateIn(t_i,\tau,r_{c_m,c_n})}{\sum_{r_{c_m,c_n}.ln_w \in LN_{r_{c_m,c_n}}} rateOut(t_i,\tau,r_{c_m,c_n}.ln_w)}$$

The environment state for each intersection controller c_n is the set of states of its incoming roads. In order to avoid the state-action dimensionality problem inherent to RL approaches, we assign the rates of *low*, *medium* or *high* to the traffic flow rates, based on their values.

Action Selection

An agent c_n's action is to assign a value to a threshold. In order to avoid the dimensionality problem, we only allow the assignment of specific values for a, d, and g. These correspond to values which are meaningful in a real-world setting and were derived from experimental data. For example, d which represents the duration that a phase should be flagged as congested in order to be considered as congested, can take the values of 25%, 50%, 75% or 99%. This means that, for the case d is given the value of 75, a controller c_n – which analyzed the state of phase $ph_{c_n,k}$ for the past x minutes and found that 75% of the time the phase was congested – will change the phase status to congested. a can take values of $\{0.5, 1, 1.5, 2\}$, and g values of $\{0.01, 0.2, 0.4, 0.6\}$.

Rewards

We consider two reward types.

Rewards for minimizing delay (r_D). This reward is used to update the Q-values of a, d and g. Reward r_D is defined as the variation in the traffic flow rate. For road segment r_{c_m,c_n} in

the time interval τ (e.g., three minutes) that ends at time t_i, $\xi_{r_{c_m,c_n}}(t_i, \tau)$ is defined as the sum of traffic flow rates of its lanes.

Intersection controller c_n computes the reward of an action act at time t_i and state s as:

$$r_D(act, s, t_i) = \xi_{r_{c_m,c_n}}(t_i, \tau) - \xi_{r_{c_m,c_n}}(t_i + \tau, \tau)$$

Rewards for controlling collaboration scope (r_C)

This reward is used together with reward r_D to update the Q-value of threshold g. Reward r_C is computed as:

$$r_C(act, s, t_i) = r_D(act, s, t_i) + (1 - 2 \times \frac{N_{plan_{new}}}{N})$$

where $N_{plan_{new}}$ is the number of intersections that are involved in the decision about the new plan and N is the total number of intersections in the network.

CHAPTER 3

CASE STUDY

To illustrate the various steps of DALI algorithms, we use a section of the City of Richardson's road network (See Figure 3.1). As shown in Figure 3.2, c_2 has four incoming roads. The four phases for c_2's intersection are $\{ph_{c_2,1}, ph_{c_2,2}, ph_{c_2,3}, ph_{c_2,4}\}$. These phases apply as follows: $ph_{c_2,1}$ for r_{c_6,c_2}, $ph_{c_2,2}$ for r_{c_3,c_2}, $ph_{c_2,3}$ for r_{c_5,c_2} and $ph_{c_2,4}$ for r_{c_1,c_2}. The phases have the following attribute values: the split $\gamma = 40$, the minimum green $\nu = 20$, the maximum green $\eta = 60$. Thresholds a, b and d have the values of $a = 0.6$, $b = 100$ and $d = 80$. In this example, c_2 evaluates the status of its intersection at the time-stamp t_{4100} and within the time interval $\tau = 500$. It starts with phase, $ph_{c_2,1}$ and calculates the average traffic throughput $Cong_{t_{4100},ph_{c_2,1}}$ for the set of road lanes that $ph_{c_2,1}$ controls. Given that $rateOut(t_{4100}, 500, r_{c_6,c_2}.ln_3) = 0.8$, $p(r_{c_6,c_2}, r_{c_6,c_2}.ln_3) = 0.2$ and $rateIn(t_{4100}, 500, r_{c_6,c_2}) = 2.4$, the value of $\xi_{t_{4100},500,r_{c_6,c_2}.ln_3}$ is:

$$\xi_{t_{4100},500,r_{c_6,c_2}.ln_3} = \frac{2.4 \times 0.2}{1} = 0.48$$

For the sake of illustration, we assume that $Cong_{t_{4100},ph_{c_2,1}} = 0.83$ which is greater than threshold $a = 0.6$. Controller c_2, then retrieves the calculated values of $Cong$ between time stamps t_{4100} and t_{4000} and finds that 91 of them are greater than a. Therefore,

$$PercentCong_{t_{4100},ph_{c_2,1}} = \frac{91}{100} \times 100 > 80$$

Consequently, c_2 detects congestion on phase $ph_{c_2,1}$ and deliberates to define a new configuration.

We assume that the average $Cong$ for phase $ph_{c_2,1}$ in the last $\nu = 10$ evaluation cycles is 0.9. Given that $e = 1$ and $f = 0.33$, c_2 defines a new configuration for $ph_{c_2,1}$, and computes $conf_{new}.phase.\gamma$ as:

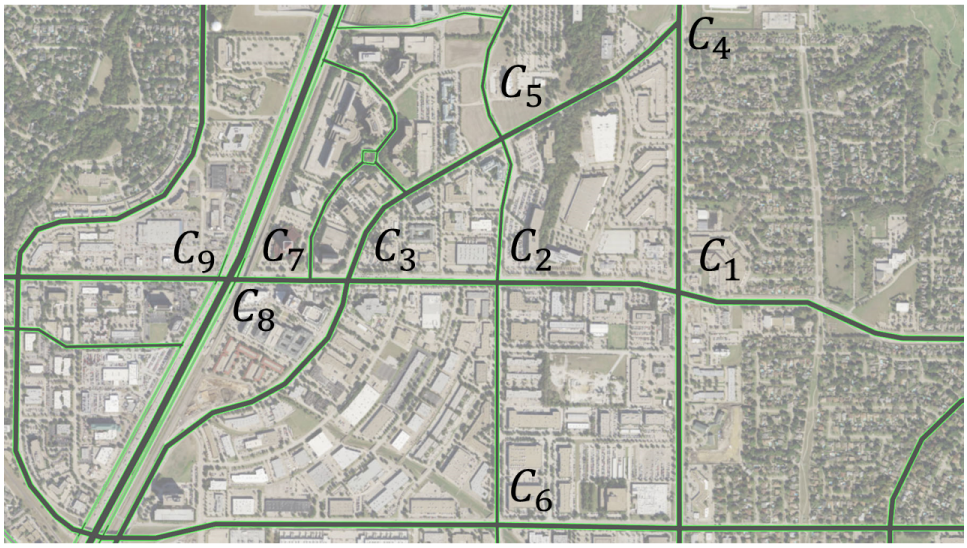

Figure 3.1. Overview of the network in the case study.

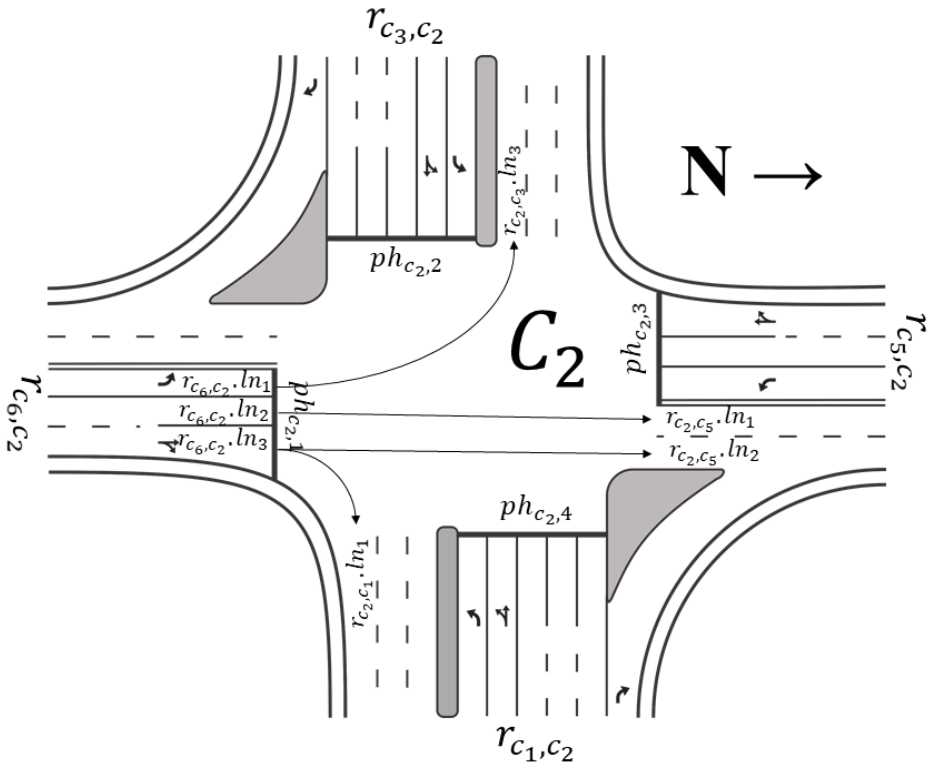

Figure 3.2. Intersection assigned to c_2.

$$conf_{new}.phase.\gamma = 40 + (1 + 0.9 \times 0.33) \approx 52$$

Therefore, c_2 determines that it needs to increase $ph_{c_2,1}.\gamma$ by 12 seconds.

c_2 proceeds by calculating $\kappa_{r_{c_1,c_2}.ln_1}$ as:

$$\kappa_{r_{c_1,c_2}.ln_3} = \frac{0.8 \times (52 - 40)}{52} = 0.18$$

Given that $\kappa_{r_{c_6,c_2}.ln_2} = 0.32$ and $\kappa_{r_{c_6,c_2}.ln_1} = 0.16$, $\kappa_{ph_{c_2,1}}$ takes the value of 0.66. Controller c_2 then calculates the effect of executing a new configuration on its neighboring intersections, including c_5. Assuming $p(r_{c_6,c_2}.ln_3, r_{c_2,c_5}.ln_2) = 0.5$, $p(r_{c_6,c_2}.ln_3,r_{c_2,c_1}.ln_1) = 0.5$, $p(r_{c_6,c_2}.ln_2,r_{c_2,c_5}.ln_1) = 1$ and $p(r_{c_6,c_2}.ln_2,r_{c_2,c_1}.ln_1) = 0$, $\kappa_{r_{c_2,c_5}}$ is calculated as:

$$\kappa_{r_{c_2,c_5}} = 0.5 \times 0.18 + 1 \times 0.32 = 0.41$$

c_2 then sends a request for evaluation to c_5 with $\kappa_{r_{c_2,c_5}} = 0.41$ and $\kappa_{ph_{c_2,1}} = 0.66$. This indicates that, by executing $conf_{new}$, an additional 0.66 vehicle per seconds (vps) will leave the road controlled by $ph_{c_2,1}$, and out of the 0.66 vps, 0.41 vps will enter r_{c_2,c_5}.

c_5 receives the request for the new timing configuration. It calculates Ψ_{c_5} using the current $rateOut(t_{4100}, 500, r_{c_2,c_5}.ln_1) = 1$, $rateOut(t_{4100},,500, r_{c_2,c_5}.ln_2) = 0.3$, $rateIn(t_{4100}, 500, r_{c_2,c_5})$- $= 1.2$, $p(r_{c_2,c_5}, r_{c_2,c_5}.ln_1) = 0.8$ and $p(r_{c_2,c_5}, r_{c_2,c_5}.ln_2) = 0.2$. Ψ_{c_5} is calculated as:

$$\begin{aligned}\Psi_{c_4} = 1.0 \times 2.0 \times \frac{0.41}{0.66} \times ((1 - 1 \times \frac{(0.41 + 1.2) \times 0.8}{1}) + \\ (1 - 1 \times \frac{(0.41 + 1.2) \times 0.2}{0.3})) \\ = -1.26\end{aligned}$$

c_5 proceeds by calculating κ for c_3, c_4. If κ is greater than threshold g, c_5 requests that they evaluate the configuration. c_3 and c_4's responses are added to Ψ_{c_5} and sent back to c_2. Upon receipt of Ψ_{c_5}, Ψ_{c_1} and Ψ_{c_3}, controller c_2 calculates Ψ_2. Negative values of Ψ are considered as a level of disagreement. Having $\Psi_2 = 2.34$, c_2 executes the new configuration and announces the execution to all controllers in the network which in turn adapt their timing configurations.

CHAPTER 4

MATISSE: A SIMULATION SYSTEM FOR AGENT-BASED INTELLIGENT TRANSPORTATION SYSTEMS[1]

In this chapter we start by giving an overview of MATISSE 2.0, a large-scale multi-agent based traffic simulation system. We proceed by discussing new features that we have developed as part of this research.

4.1 Overview of MATISSE 2.0

4.1.1 High Level Architecture

MATISSE's architecture consists of three building blocks. The simulator's main constituent is the *Simulation System* which includes three subsystems:

1. The *Agent System* creates and manages simulated standard and ATS-enabled vehicles and intersection controllers as well as zone managers. The various agent types communicate through the Agent-to-Agent Message Transport Service;

2. The *Environment System* creates and manages the traffic network;

3. The *Simulation Microkernel* manages the simulation workflow.

The *Control and Visualization System* renders 2D and 3D representations of the simulation and provides real-time interaction mechanisms. The *Message Transport Service* provides a configurable messaging infrastructure that allows MATISSE's building blocks to exchange information.

[1] ©2018 IEEE. Portions Adapted, with permission, from Behnam Torabi, Rym Z. Wenkstern, and Mohammad Al-Zinati. "An Agent-Based Micro-Simulator for ITS." In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, IEEE ITSC 2018, November 2018.

4.1.2 Traffic Network Structure

In MATISSE, a traffic network is specified as a directed graph where nodes represent intersections or connections between road segments, and directed edges represent road segments. The graph defines the possible traffic movements between lanes in consecutive roads. Figure 4.1 (a) shows the graph definition that corresponds to the traffic network illustrated in Figure 4.1 (b).

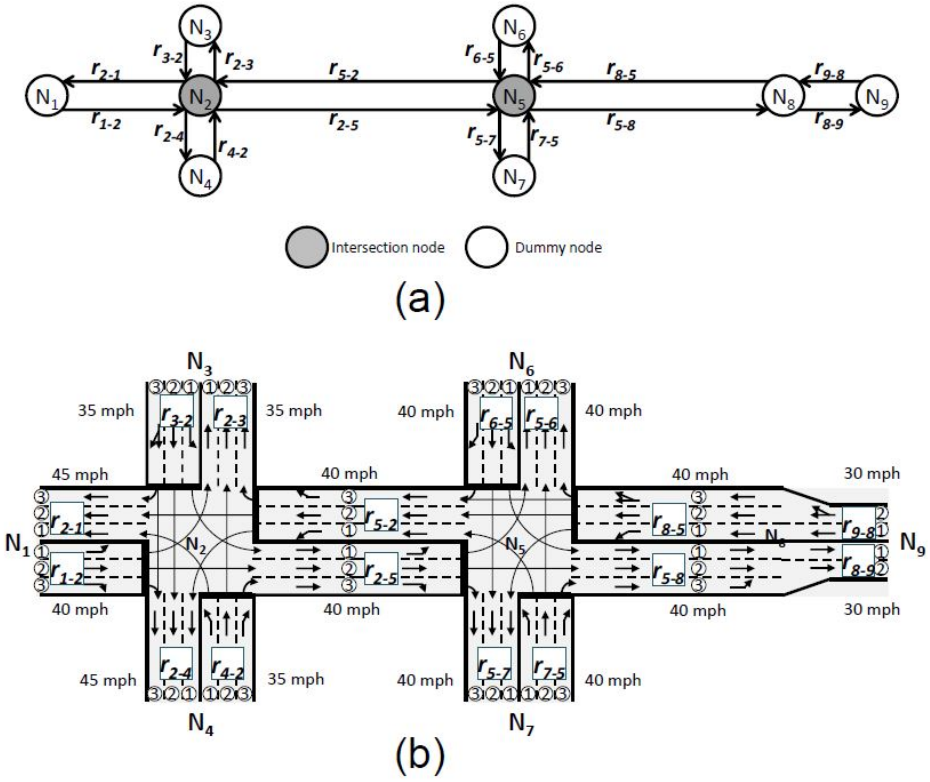

Figure 4.1. Traffic Network Definition in MATISSE

4.1.3 Creating Virtual Agents

MATISSE was built as a multi-agent simulation system from the ground up. It provides several predefined concrete classes for vehicle agents. These classes can be instantiated to

create various types of virtual traffic agents equipped with diverse sensing and communication mechanisms. The modeler can also easily create new types of agents by implementing concrete classes extending from abstract classes provided in MATISSE's *vehicle agent package*. The vehicle agent package consist of four main modules:

1. *Interaction module*: This module handles the vehicle agent's interaction with external entities. It consists of three sub-modules: a) The *perception module* implements mechanisms necessary for a vehicle agent to perceive the traffic environment using sensors; b) The *communication module* handles all communications between the vehicle agent and other simulated agents; and c) The *route guidance module* implements mechanisms that allow a vehicle agent to access the traffic network information (e.g., route guidance, congestion levels on roads).

2. *Knowledge module*: This module represents the vehicle agent's memory. It includes the vehicle agent's knowledge about itself (e.g., acceleration, deceleration capabilities, maximum speed) and knowledge acquired through sensing and communication (e.g., approaching vehicle).

3. *Task module*: This module defines the tasks that a vehicle agent can perform. *MoveTask* and *TurnTask* are used by the vehicle to travel in the environment.

4. *Planning module*: This module implements different vehicle agent planning strategies. The *TravelRoutePlanningModule* implements plans used by the vehicle to find a travel route, while the *MovementDynamicsPlanningStrategy* computes a set of possible actions, their reward and risk values and selects an action based on the driving behavior.

Figure 4.2 shows a section of the code for a vehicle agent. A demo illustrating MATISSE's features is available at: *mavs.utdallas.edu/its*

```
public class VehicleAgent

extends AbstractAgent<VehicleAgentState, VehicleKnowledgeModule, VehicleInteractionModule,
    VehiclePlanningModule, VehicleTaskModule>
implements Serializable
{
    public VehicleAgent(VehicleAgentState state, CellBounds cellBounds) {
        super(state);
    }
    @Override
    protected VehicleInteractionModule createInteractionModule(VehicleKnowledgeModule knowledgeModule) {
        return new VehicleInteractionModule(new VehiclePerceptionModule(knowledgeModule),
                new SimpleAgentCommunicationModule(knowledgeModule.getId()));
    }
    @Override
    protected VehicleKnowledgeModule createKnowledgeModule(VehicleAgentState state) {
        return new VehicleKnowledgeModule(state);
    }
    @Override
    protected VehiclePlanningModule createPlanningModule(VehicleKnowledgeModule knowledgeModule,
            VehicleTaskModule taskModule, VehicleInteractionModule interactionModule) {
        VehiclePlanGenerator planGenerator =
                new VehiclePlanGenerator(knowledgeModule, taskModule, interactionModule);
        VehiclePlanExecutor planExecutor =
                new VehiclePlanExecutor(knowledgeModule);
        return new VehiclePlanningModule(planGenerator, knowledgeModule, planExecutor);
    }

    @Override protected VehicleTaskModule createTaskModule(VehicleKnowledgeModule arg0) {
        return new VehicleTaskModule(arg0);
    }
```

Figure 4.2. Code for vehicle Agent

4.2 MATISSE 3.0

In this section, we discuss the features that we added to MATISSE 2.0 to allow the implementation and the validation of DALI.

4.2.1 Importing Traffic Networks from OpenStreetMap

In MATISSE 3.0 ((Torabi et al., 2018) and (Torabi et al., 2018)), the modeler can either create a virtual traffic network through a graphical interface by "snapping" road segments or by importing entire networks through Open Street Map (OSM). The modeler can also import a section of an OSM network from a file system or an online viewer. Several advanced algorithms have been developed to reliably convert OSM graphs to MATISSE graphs, and automatically generate missing information, e.g., number of road lanes, traffic light locations, and allowable traffic movements. In this section, we give an overview of OSM network structures and some of MATISSE's conversion algorithms.

OpenStreetMap Network Structure

OSM networks are in the form of XML formatted files. They contain three types of elements: a) *node*, b) *ways* and c) *relations*. Elements can have *tags* and *keys* to describe their features. A *node* holds coordinates of a location. A *way* is an ordered list of *nodes*. A *way* can either be *open* or *closed*. A *way* is closed if it has the same first and last *nodes*. It is open otherwise. *Relations* are included in an OSM file to describe the relations between *ways* and *nodes*. In OSM file format, *ways* that are marked as "highway" represent roads.

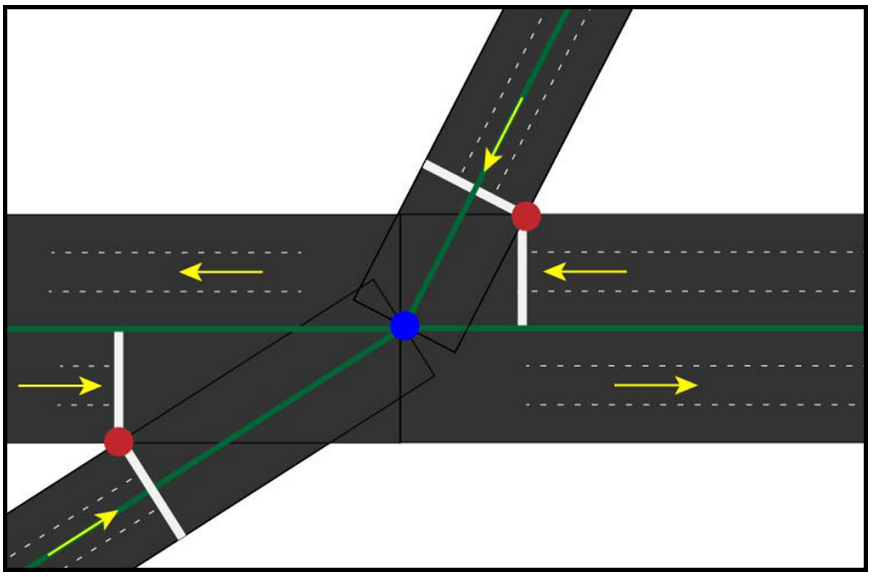

Figure 4.3. Generate an intersection from an OSM file

Converting OpenStreeMap Data into MATISSE's Networks

To convert an OSM network, MATISSE first reads the OSM file and extracts *ways* that represent roads. Then, it adds nodes associated with roads into its network graph. Next, it connects the nodes in the order that is specified in the OSM file. Then, it widens the roads depending on their number of lanes. The number of lanes is usually specified in OSM files. In case it is undefined, MATISSE estimates it based on the road type.

In certain cases, widening roads creates overlaps between road surfaces at intersections (see Figure 4.3). For these cases, stop bar positions have to be computed. MATISSE places a stop bar at the position where a road surface crosses the surface of its adjacent roads. In figure 4.3, red circles show crossing points.

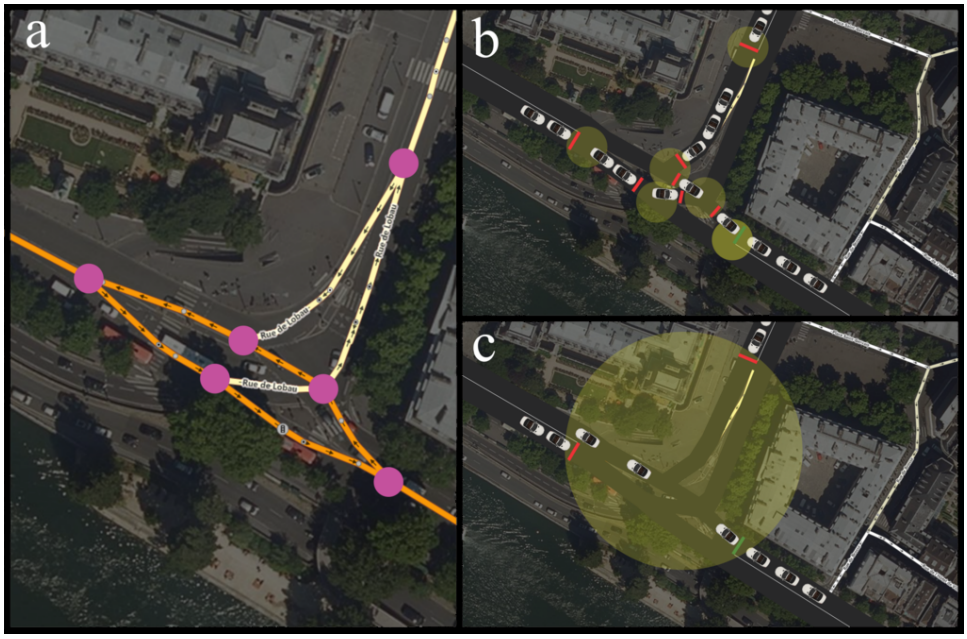

Figure 4.4. An Intersection in Paris and Its Representation in Open Street Map

After forming intersections, signalized intersections need to be determined. Most micro-simulators represent a signalized intersection with one node in their network structure. However, in Open Street Map a signalized intersection is not necessarily represented by one node, but often with an arbitrary number of nodes. Figure 4.4(a) shows an OSM graph for a complex signalized intersection in Paris, France. The single signalized intersection is represented using 6 nodes. Micro-simulators such as SUMO or VISSIM convert each of the OSM nodes into one signalized intersection which results in an incorrect representation of the real network topology (see Figure 4.4(b)). As shown in Figure 4.4(c), MATISSE's network structure and conversion algorithms allow an accurate conversion of the information.

4.2.2 Defining Vehicle Distribution at Initialization Time

At the start of the simulation, the user defines the total number of vehicles to be run as well as entry and exit points. The initial vehicle distribution can be automatically generated by MATISSE or specified by the user through a graphical interface.

4.2.3 Defining New Vehicle and Intersection Controller Behaviors

Virtual ATS-enabled vehicles and intersection controllers are equipped with sensors and perceive the environment within their sensor range, called *circle-of-influence* (COI). They are able to communicate with other enabled vehicles and intersection controllers located within their COI (see Figure 4.5(a)).

For standard virtual vehicles, a *vision cone* is used to simulate a human driver's vision range and perception of the environment. Other virtual human sensors such as auditory and olfactory sensors are available. The virtual driver's *level of distraction* is directly related to its level of perception of its direct environment (See Figure 4.5(b)).

The ranges of the various sensors can be altered during the execution of simulation.

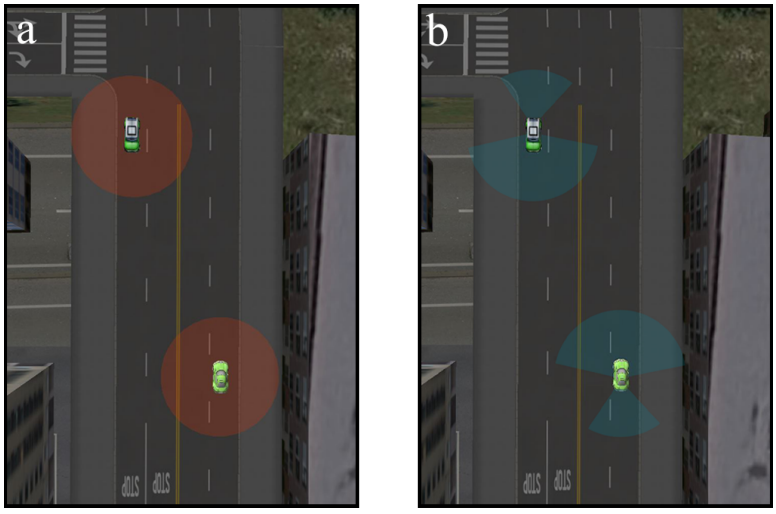

Figure 4.5. Vehicle agents perception. a) Circle of Influence b) Vision cones

Vehicle Behavior

Unlike most simulators which use predefined car-following and lane-changing algorithms, in MATISSE 3.0, vehicle agents compute the set of possible actions based on their current perception of the environment, the maximum acceleration and deceleration rates and their flexibility in steering. They assign a reward and a risk value to each action. The reward indicates the impact that an action has in helping the vehicle achieve its objectives (e.g., reach its destination as fast as possible, follow traffic rules, drive smoothly). For each action, the agent also assesses the risk that an action may result in a collision. For normal driving behavior, the agent ignores the actions that are considered dangerous and executes the action with the highest reward.

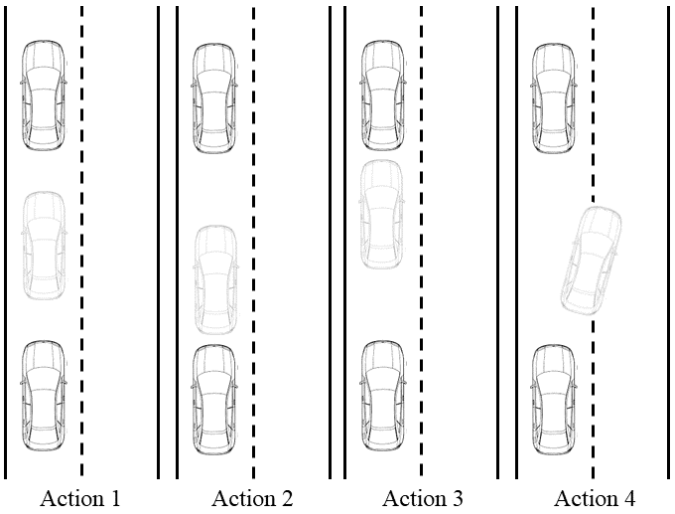

Figure 4.6. Four possible actions of a vehicle

Various driving behaviors can be simulated by assigning different values to the risk-aversion factor and the importance of an objective in the reward values. Figure 4.6 shows a scenario where the vehicle agent in the back can take different actions. In *Action 1* the vehicle agent maintains its current speed and steering. In *Action 2* it maintains current steering and decreases its speed. In *Action 3* it maintains current steering and increases its

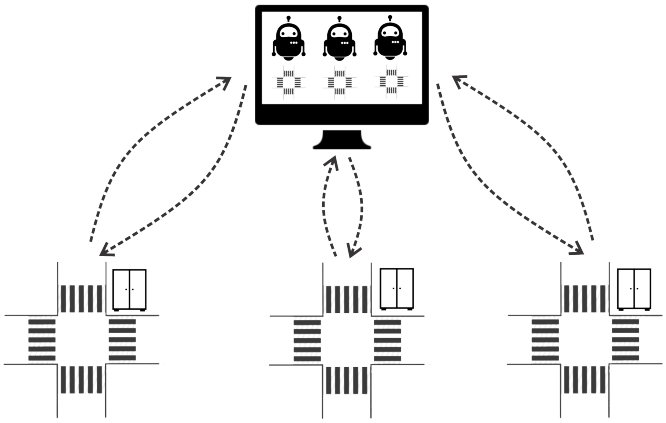

Figure 4.7. Architecture of the Hybrid Simulation.

speed. Finally, in *Action 4* it maintains its current speed and steers the vehicle 20 degrees to the right. If we assume that with respect to the vehicle agent's objectives, *Actions 3* has the highest reward and *Action 4* has the lowest reward, and with respect to collision, *Action 3* has the highest risk and *Action 2* has the lowest risk then, among the four actions, an aggressive agent will choose *Action 3*, a defensive agent will choose *Action 2*, a regular agent will select *Action 1*, and a careless agent will choose *Action 4*.

4.2.4 Hybrid Simulation

We implemented a feature that allows MATISSE to connect to deployed controllers in order to obtain real-world traffic information in real-time. Architecture of the Hybrid simulation can be seen in Figure 4.7. The City of Richardson's intersection controllers integrate web servers which are used by traffic engineers to connect remotely to the controllers. MATISSE connects to the controllers' web servers through VPN. It sends a request for an update on the status of the intersections every three hundred milliseconds. The controllers respond by sending the current state of their traffic lights (i.e., red, green or yellow) and detectors (i.e., active or inactive).

CHAPTER 5

EVALUATION OF DALI'S SIMULATION RESULTS[1]

DALI's algorithms were implemented in MATISSE 3.0. In this chapter we discuss the experimental results for the execution of DALI on a simulated model of the City of Richardson.

5.1 Metrics

The common evaluation metrics for the performance of traffic signal timing systems is *delay*. Delay is defined as the increment in a vehicle's travel time caused by traffic control devices, compared with the travel time if the vehicle was to maintain its expected speed in the absence of any control device (Balke and Herrick, 2004).

5.2 Simulation Setting

The experiments were run on a multicore PC (Intel Core i7 X980 CPU (3.33GHz), 6.00 GB, 64-bit Windows 7). A simulated model of the City of Richardson's road network was created in MATISSE. The model includes 1365 road segments and the city's 128 signalized intersections in addition to the 965 non-signalized intersections. Figure 5.1 shows a 2-D representation of the traffic network. Tables 5.1 and 5.2 summarize the types of signalized and non signalized intersections, classified based on the number of incoming and outgoing lanes.

Three simulation settings were run eight times for 86,400 simulation cycles representing a 24-hour time period.

---

[1] ©2018 IEEE. Portions Adapted, with permission, from Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. "A Collaborative Agent-Based Traffic Signal System for Highly Dynamic Traffic Conditions." In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, IEEE ITSC 2018, November 2018.

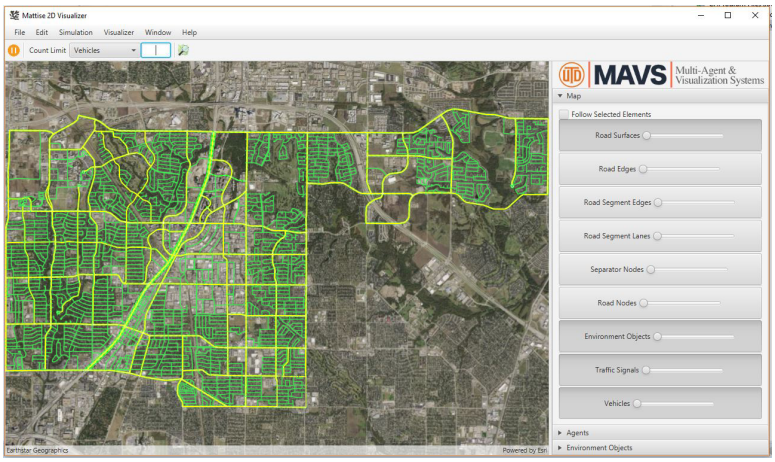

Figure 5.1. 2D visualization of Richardson's Traffic Network.

Table 5.1. Number of Signalized Intersection with various incoming and outgoing lanes

Type	1×1	1×2	1×3	2×2	2×3	3×3
Count	0	4	8	18	29	69

Table 5.2. Number of Non-Signalized Intersection with various incoming and outgoing lanes

Type	1×1	1×2	1×3	2×2	2×3	3×3
Count	533	241	175	16	0	0

5.3 Experiment 1: DALI with fixed threshold values

In the first and second experiment, we use real-world data provided by the City of Richardson to simulate regular traffic patterns with and without accidents. In the third and fourth experiment we simulate continuous random traffic patterns with and without accidents. For all experiments, we compare the efficiency of DALI with the SCATS-based system currently in use in Richardson (SCATS-R), and a model of the RL-based MARLIN-ATSC (El-Tantawy et al., 2013) (MARLIN-R). To decrease the learning time of MARLIN agents, we initialized the Q-values based on estimations derived from historical data provided by the City of Richardson.

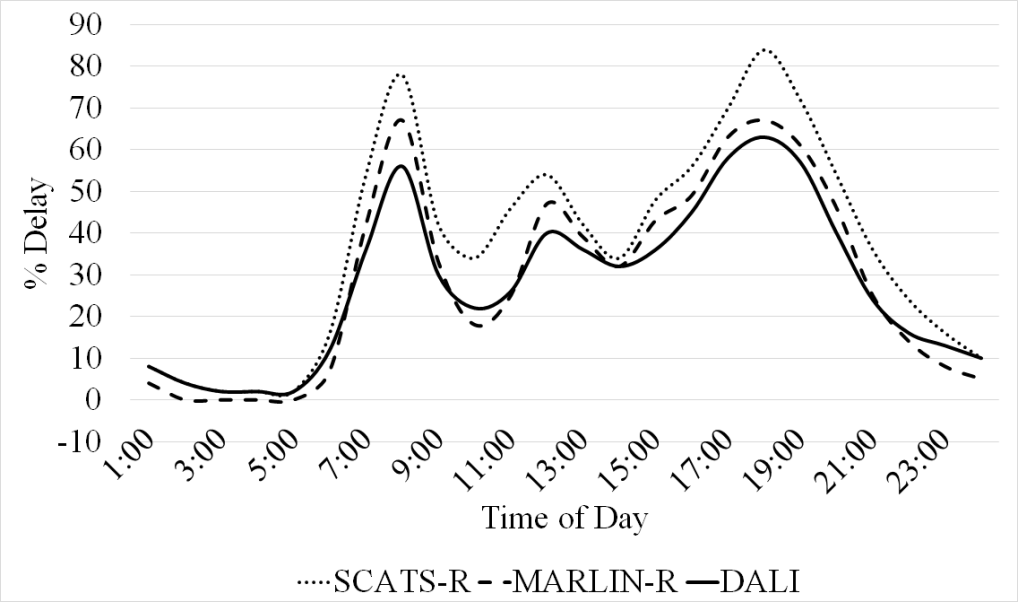

Figure 5.2. Average delay using traffic data from the City of Richardson

5.3.1 Experiment 1.1: Normal Traffic Conditions

In this experiment, we make use of the traffic data provided by the City of Richardson to determine the number of vehicles in the traffic network at any given time, as well as their distribution in the network. This experiment is intended to analyze the behavior of the three systems under nominal traffic conditions.

As shown in Figure 5.2, between the times of 00:30 am and 5:30 am DALI and SCATS-R perform at the same level with respect to delay. This is due to the fact that during this time period, traffic is very light and therefore DALI agents do not perform any action. MARLIN-R agents perform better (53% delay reduction) in this situation because of their flexibility in changing the traffic phases at any time. As we progress during the day (i.e., 6:30 am to 8:30 am) the traffic flow increases, and congestion is detected. DALI agents naturally collaborate with one another to define and implement timing plans that meet the network conditions. As such, DALI performs significantly better than SCATS-R (23% delay reduction).

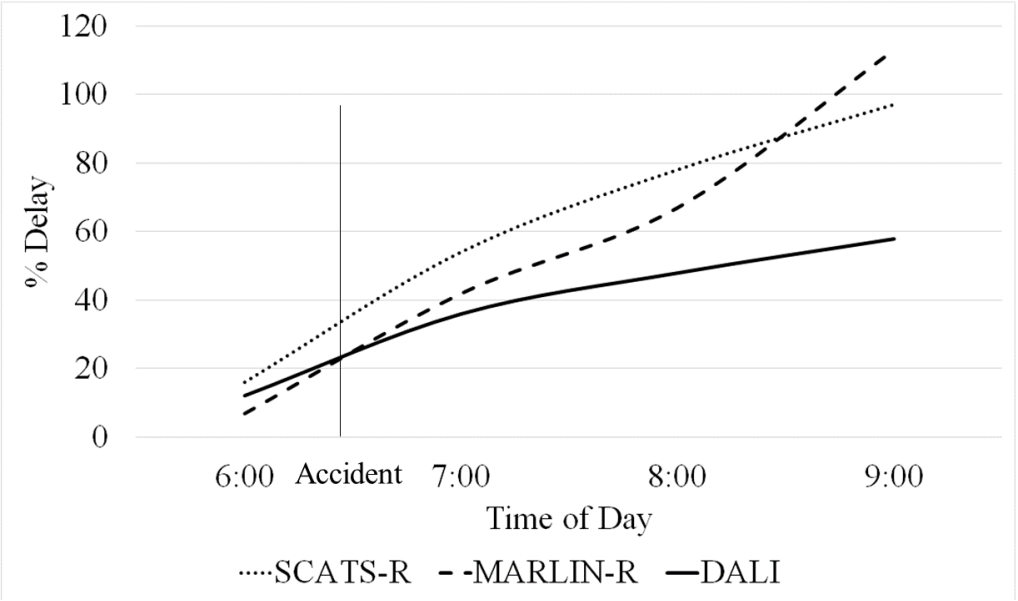

Figure 5.3. Average delay with accident in peak morning hours using real traffic data

MARLIN-R performs slightly less than DALI. The simulation shows that this is due to the fact that MARLIN-R agents do not handle heavy traffic in small network areas with a large number of intersections efficiently. In those cases, MARLIN-R agents give the right-of-way to vehicles without taking into account the downstream roads which are congested.

5.3.2 Experiment 1.2: Normal Traffic Conditions With Accident

Figure 5.3 shows the performance of the systems when an accident is triggered at run time, during normal morning peak traffic. As expected, DALI handles the traffic much better than SCATS-R (35% delay reduction). MARLIN-R agents are unable to control the congestion created by the accident since they have no prior knowledge of the unexpected traffic pattern. Similarly to Experiment 1, the simulation shows that, rather than leading the vehicles towards roads with lighter traffic, MARLIN-R agents send vehicles to congested areas.

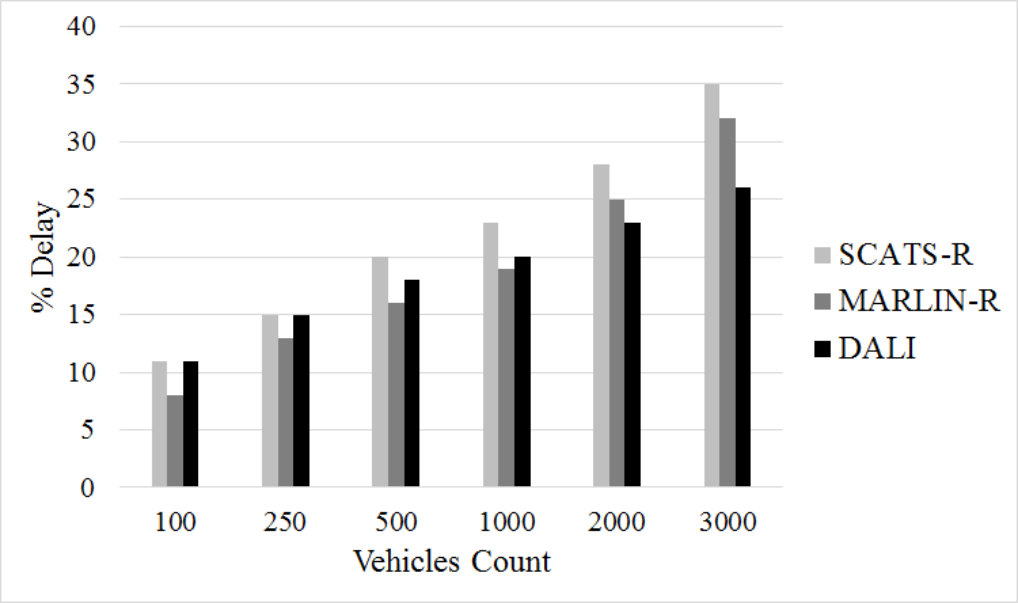

Figure 5.4. Average delay for random traffic patterns

5.3.3 Experiment 1.3: Continuous Random Traffic Conditions

In this experiment, the number of vehicles during the simulation remains constant but new vehicles are added randomly while others randomly exit the traffic network. This experiment is intended to illustrate random traffic patterns that are unprecedented. The experiment was run with 100, 250, 500, 1000, 2000 and 3000 vehicles.

Figure 5.3 shows that when the traffic is light, MARLIN-R agents perform (37%) better because they use a variable phasing sequence. They can extend the current phase or Result-Four to any other phase according to the changes in traffic. On the other hand, SCATS-R controllers and DALI agents execute a fixed phase sequence. Therefore, all phases are executed even in cases where it is not necessary. DALI and SCATS-R perform at the same level in lighter traffic conditions because the controller agents do not detect congestion and therefore, do not change the split. As the number of vehicles increases, DALI agents start to detect congestion and collaborate with other agents for retiming. The collaborative retiming procedure allows DALI to perform better than SCATS. As the number of vehicles increases,

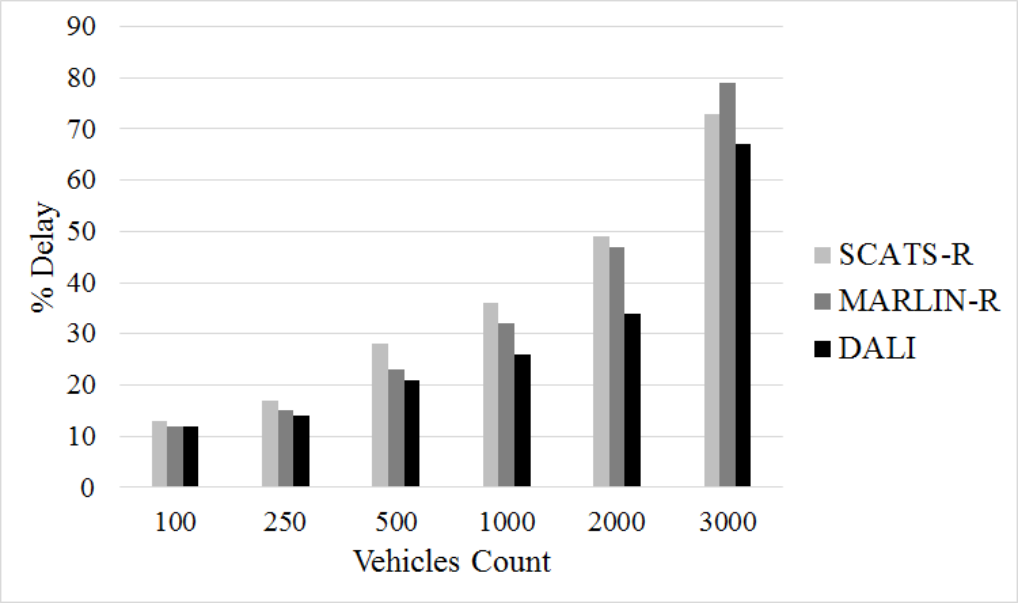

Figure 5.5. Average delay for random traffic patterns with accidents.

MARLIN-R still perform better than SCATS-R. However, DALI performs better. This is due to the fact that MARLIN-R agents fail to handle heavy traffic in small, condensed network areas.

5.3.4 Experiment 1.4: Continuous Random Traffic Conditions with Accident

Figure 5.5 shows the performance of DALI, SCATS-R and MARLIN-R in the extreme situation where an accident is randomly triggered in unpredictable traffic conditions. When the traffic is light, the three systems nearly act the same. As traffic gets heavier, DALI operates better than the other two (20% decrease in delay compared to SCATS-R and 12% decrease in delay compared to MARLIN-R). When the number of vehicles reaches 3000, MARLIN-R operates worse that SCATS-R (8% delay increase) because SCATS-R controllers are committed to giving green signal to all movements in a cycle whereas MARLIN-R agents lack experience in dealing with new traffic conditions.

5.4 Experiment 2: DALI with Adaptive Threshold Values

In this experiment, we use real-world data provided by the City of Richardson to simulate regular traffic patterns. We compare the efficiency of DALI with fixed threshold values of $a = 0.5$, $d = 75$ and $g = 0.4$, the SCATS-based system currently in use in Richardson (SCATS-R), DALI with adaptive threshold values (DALI-RL), and a model of the RL-based MARLIN-ATSC (El-Tantawy et al., 2013) (MARLIN-R). Both DALI-RL and MARLIN-R Q-values were initialized based on estimations derived from historical data provided by the City of Richardson.

5.4.1 Experiment 2.1: Assessing Delay

As shown in Figure 5.6, between the times of 00:30 am and 5:30 am, DALI and SCATS-R perform at the same level with respect to delay. This is due to the fact that during this time period, traffic is very light and therefore DALI agents do not perform any action. As expected, DALI-RL performs better (21% delay reduction) in comparison with DALI and SCATS-R. MARLIN-R agents also perform better (53% delay reduction) than DALI because of their flexibility in changing the traffic phases at any time. As we progress during the day (i.e., 6:30 am to 8:30 am) the traffic flow increases, and congestion is detected. DALI agents naturally collaborate with one another to define and implement timing plans that meet the network conditions. As such, DALI performs significantly better than SCATS-R (23% delay reduction). DALI performs slighly better than MARLIN-R (4% delay reduction). The simulation shows that this is due to the fact that MARLIN-R agents do not handle heavy traffic in small network areas with a large number of intersections efficiently. In those cases, MARLIN-R agents give the right-of-way to vehicles without taking into account the downstream roads which are congested. DALI-RL agents perform better in comparison with DALI agents (7% delay reduction) by adaptively selecting threshold values.

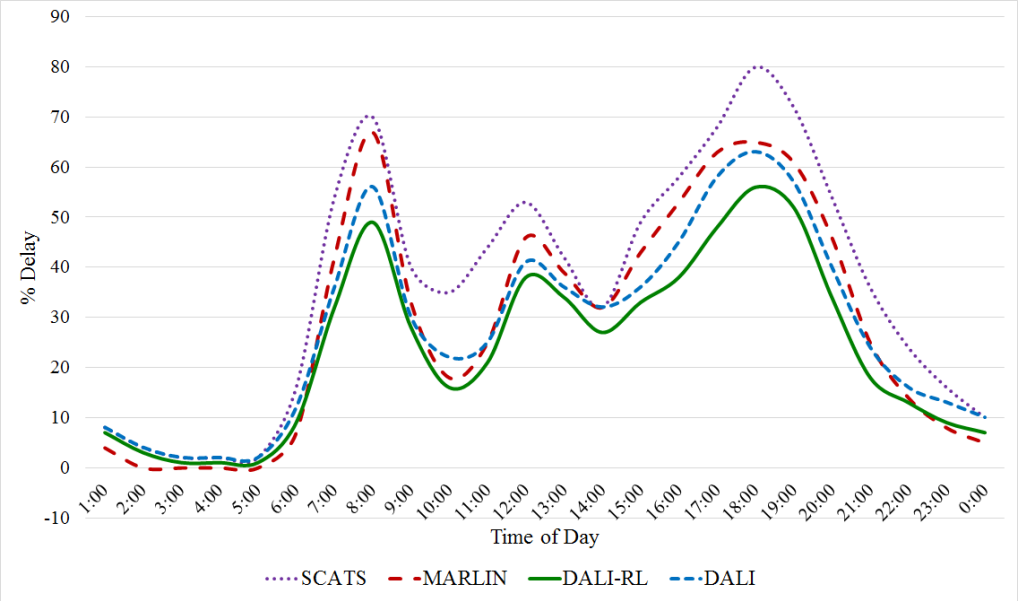

Figure 5.6. Average delay using traffic data from the City of Richardson

5.4.2 Experiment 2.2: Assessing Changing Values of Threshold a

Figure 5.7 compares the performance of DALI with MARLIN-R and SCATS-R for different values of a. For lower values of a, agents almost continuously collaborate to adapt their traffic signals. This results in lower average delay since agents do not wait until higher levels of congestion are reached to act. Nevertheless, as shown in Table 5.3, lower values of a result in a very large number of exchanged messages. Higher values of a decrease requests for retiming and consequently the average delay is increased. As shown in Figure 5.7 and Table 5.3, the adaptive selection of a allows DALI-RL agents to perform better for both average delay and number of message exchanges.

5.4.3 Experiment 2.3: Assessing Changing Values of Threshold g

Figure 5.9 shows the average size of groups that are formed dynamically when a re-timing is called for by a controller agent.

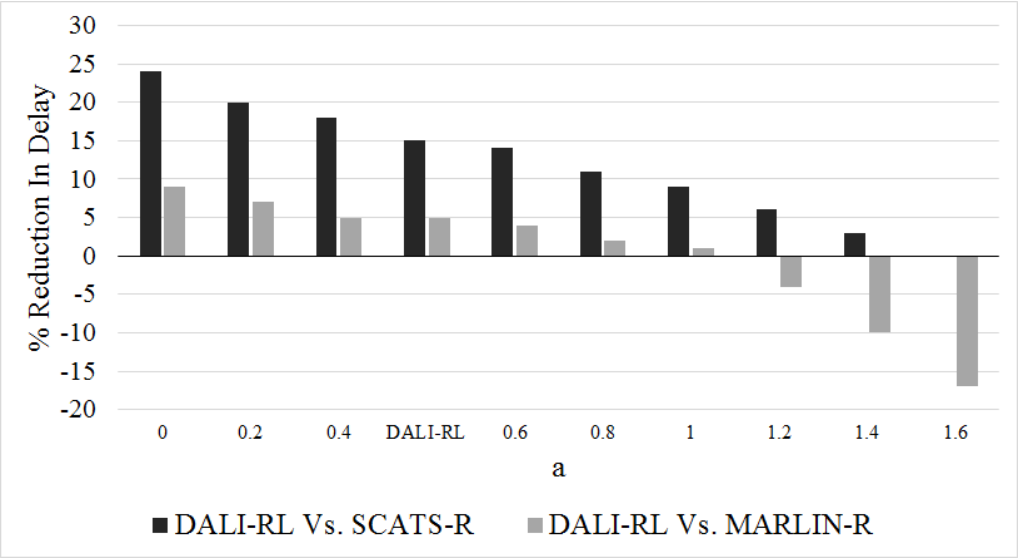

Figure 5.7. Average Delay For Different Values of a.

Table 5.3. Number of Message Exchanges For Different Values of a.

Value of a	Number of Exchanges
0.0	7,155,289
0.2	358,401
0.4	156,272
0.6	95,478
DALI RL	30,409
0.8	17,654
1.0	7,689
1.2	4,859
1.4	468
1.6	200

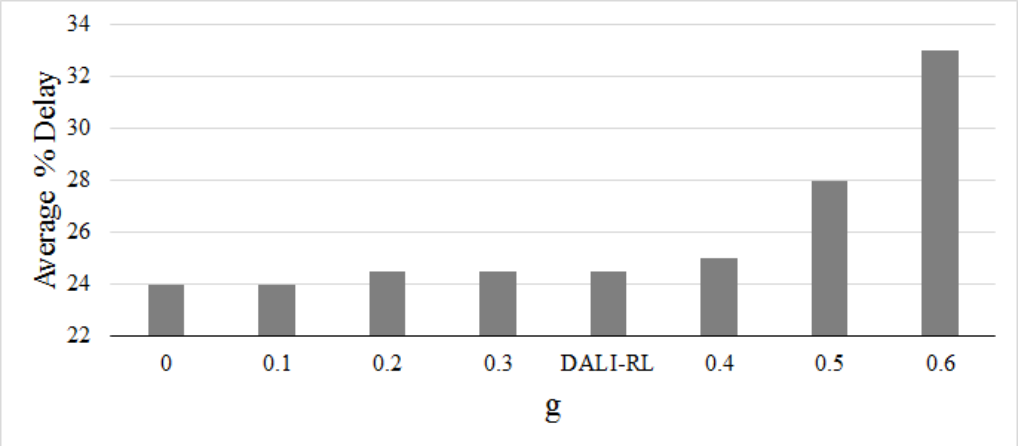

Figure 5.8. Average delay For Different Values of g.

When g is equal to zero, the propagation of requests does not stop, and therefore all the controller agents end up being involved in the collaborative re-timing process.

As g increases, the average group size becomes smaller and therefore fewer communications are needed. Using a fixed value for g is not always efficient because in certain unexpected circumstances it may be better to increase the collaboration scope. Figures 5.8 and 5.9 show that when agents use RL to determine g values, better performance is achieved with fewer communications due to the smaller group size. As illustrated in Figure 5.8, when the value of g is less than 0.4, no significant improvement occurs with respect to average delay. This is explained by the fact that, broadening the collaboration scope to include agents that are not impacted by the plan does not have any effect on the final outcome.

5.5 Experiment 3: Hybrid Simulation

As mentioned in Section 4.2.4, MATISSE is able to run hybrid simulations by retrieving real-time data from deployed controllers. This data which includes the detector states (i.e., active, inactive) and the traffic light state (i.e., green, yellow, red) is processed as follows: when a detector state goes from active to inactive, MATISSE adds a vehicle in the simulation at the detector's position. It also visualizes the queue length at the traffic light.

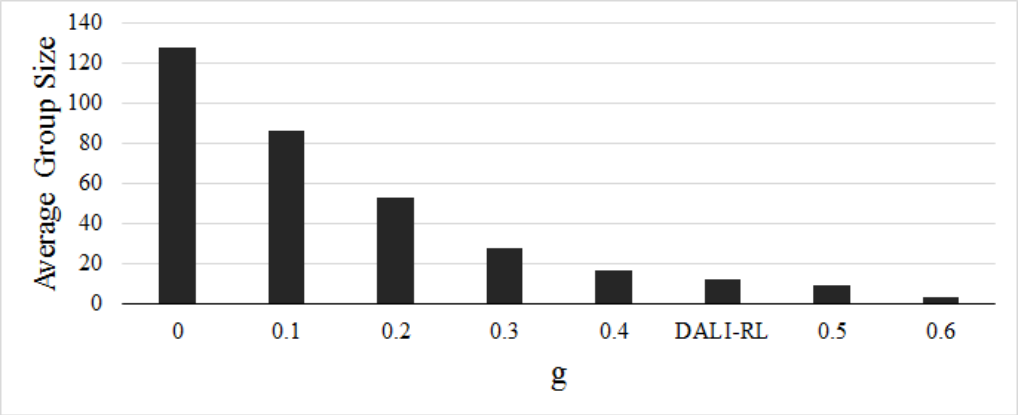

Figure 5.9. Average Group Size For Different Values of g.

In the hybrid simulation, simulated vehicles enter the simulation through entry points (represented as red arrows in Figure 5.10), and leave the simulation when they reach the exit points (represented by green arrows in Figure 5.10). The destinations of the simulated vehicles at the entry points are estimated based on the traffic flow information at the exit points. The performance of DALI in the simulated environment is evaluated by comparing the rate of simulated vehicles that exit the simulated traffic network versus the real world.

In this experiment, we create a simulated model of the Waterview corridor in Richardson, which includes three intersections, Frankford Rd, Synergy Pkway and Franklin Jenifer (see Figure 5.10). We connect MATISSE to the three real-world controllers, run the hybrid simulation using DALI and compare the results with the actual SCATS-R-based values provided by the controllers.

We ran the hybrid simulation for one week and compared the average queue length and delay in the simulation with their actual SCATS-R counterparts. Figure 5.11 shows the average traffic flow at different times of the days for network entrance points. As expected, Waterview Parkway gets a rush in the morning from 7:00 AM to 9:00 AM and in the evening from 4:00 PM to 6:00 PM. Waterview approaches have approximately the same traffic flow during the day; however, traffic drops at nights.

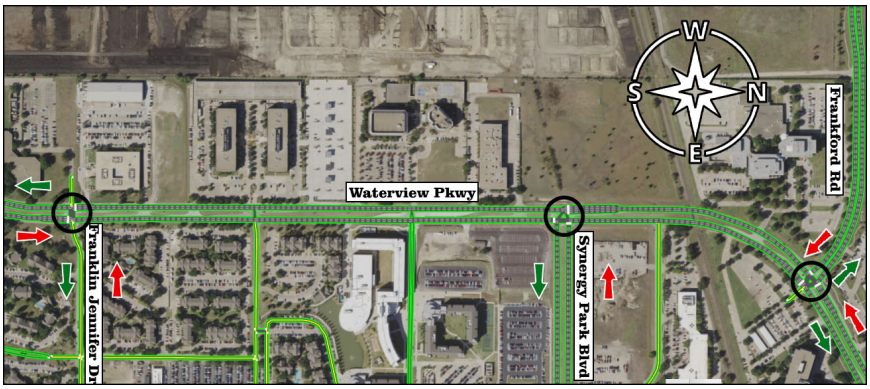

Figure 5.10. Arterial image of simulated intersections.

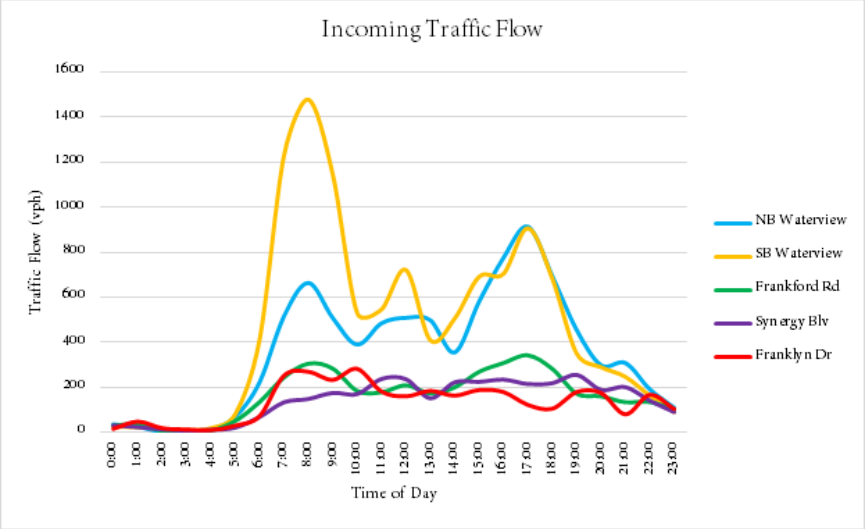

Figure 5.11. Average incoming traffic flow at different times of work days.

Table 5.4 shows the reduction in delay for different traffic flows. Similar to the previous experimental results, when the traffic is light, the average delay does not change since DALI agents do not perform any action. When the traffic flow increases, agents adapt by generating new plans and executing them. Therefore, the traffic on roads with higher demand get more green which results in a decrease of the average delay.

Figure 5.12 shows the reduction in queue length for different flow rates at different entry points. As illustrated, the queue length was drastically reduced in both directions at Waterview. However, at the same time, the queue length increased on approaches. The reason is

Table 5.4. Reduction in Delay for Different Traffic Flows

Traffic Flow (vph)	$0-200$	$200-400$	$400-600$	$600-800$
% Reduction in Delay	0	0	1.27	4.12
Traffic Flow (vph)	$800-1000$	$1000-1200$	$1200-1400$	$1400-1600$
% Reduction in Delay	7.15	10.79	17.96	20.81

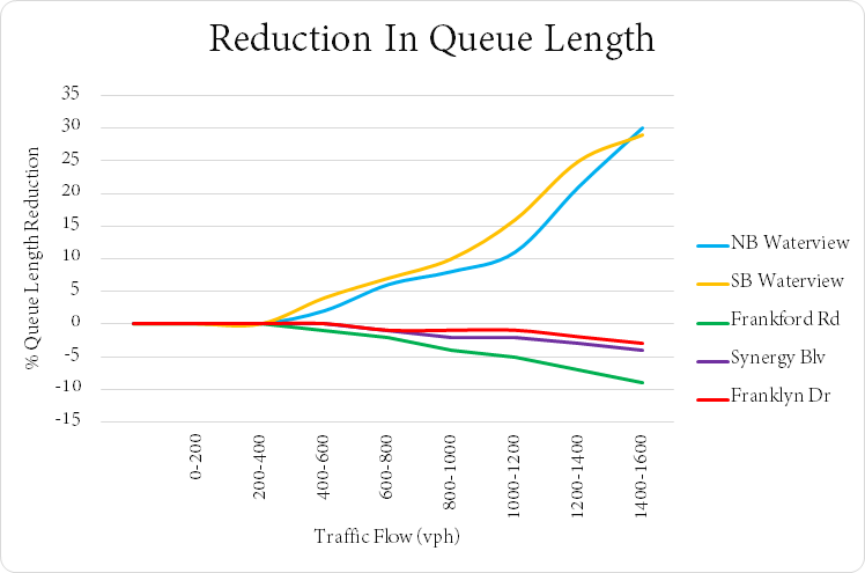

Figure 5.12. Delay Reduction For different traffic flows at different entrances.

that whenever the traffic flow increases on Waterview, agents react by increasing the green time of the phases that control Waterview. Therefore, the vehicles in the other directions receive less green time.

CHAPTER 6

DEPLOYMENT OF DALI

In this chapter we discuss the deployment of DALI on the Waterview corridor in the City of Richardson, and share the lessons learned during deployment.

6.1 The Waterview Corridor

DALI's agents were deployed on the Waterview corridor which includes three intersections (see Figure 6.1): Waterview and Frankford Rd (FO), Waterview and Synergy Park Blvd (SY), and Waterview and Franklin Jenifer Dr (FER). Figure 6.2 shows the location of the detectors at these intersections.

The phase numbering for the Waterview intersections follows the standard phase numbering discussed in Section 1.3. Phases 6, 2, 5 and 1 are assigned respectively to the northbound, southbound, southbound left turns, and northbound left turns of Waterview Pkwy. For the Frankford intersection, phase 8 is assigned to the westbound of Frankford Rd. Phase 4 is assigned to Synergy Park Blvd at the Synergy intersection and to Franklin Jenifer Drive at the Franklyn intersection. Phases such as 7 and 3 are not considered given the low traffic on the roads they control.

6.2 Safety and Monitoring Requirements

When deploying a new signal timing system, the first and most crucial issue to address is safety. Given the strict safety regulations in the US, it was necessary to devise a scheme to ensure that DALI's operations comply with these regulations. Rather than replacing the Intelight controllers (which already implement all required safety features), we used the controllers to execute the agents' decisions and pass vehicle detection data to the agents. When operating in SCATS-R mode, the Intelight controllers execute the timing plans programmed by the City of Richardson's traffic engineers. When in DALI mode, they operate

Figure 6.1. Overview of The Corridor.

as instructed by the agents while ensuring that the safety requirements are met. In case of disruption, the agents give back control to the Intelight controllers which then operate in SCATS-R mode.

While it would have been possible to place the agents inside the Intelight controller cabinets, we decided to install them in the lab computers and use VPN to connect to the controllers. This approach allowed us to closely monitor all agents concurrently and adjust their behavior in a timely manner (see Figure 6.3).

6.3 Agent Implementation

The agents were implemented in JAVA. Each agent runs on a PC with 2 gigabytes of ram and 3.33 GHz clock and communicates with the other agents through a Broadcom gigabit ethernet net link with a minimum speed of 100 mbps. Each agent runs three threads to update its intersection status, one thread to execute its algorithm and one thread to communicate with other agents. The agents' architecture is discussed in section 4.1.3. In order to implement the mechanisms that allow the agents to control and release control of the Intelight controllers, it was necessary to learn how to operate the controllers.

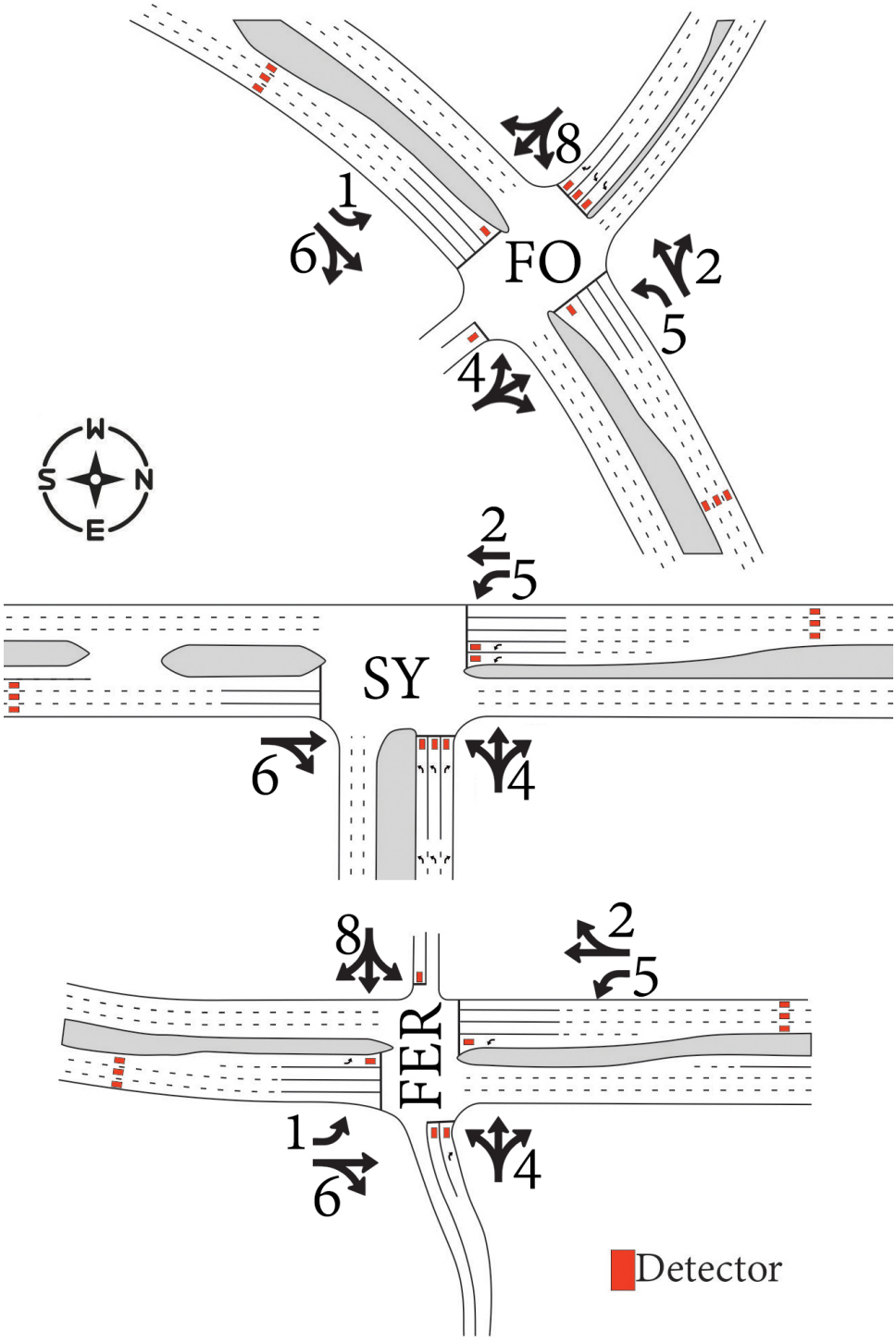

Figure 6.2. Overview of Intersections.

Figure 6.3. DALI Agents Running in the Lab.

a) Phase Status

Phase Group	1	2	3	4	5	6	7	8
Reds		●	●	●	●			●
Yellows								
Greens	●					●		
Veh Calls		🚗	🚗			🚗		🚗
Dont Walks		✋		✋		✋		✋

b) Action Parameters

Action	Pattern
1	Pattern 1
2	Pattern 2
3	Pattern 3
4	Pattern 4
5	Pattern 5

Figure 6.4. a) Status of the intersection in MAXTIME. b) Actions in MAXTIME

6.4 Intelight Controllers

An Intelight controller integrates a web server called MAXTIME which is accessed remotely via VPN. The communication protocol with the client side is achieved through html forms.

a) **Day Plan Events**

Event	Hour	Minute	Action
1	6	0	1
2	8	30	2
3	11	0	3
4	16	30	4
5	18	0	3
6	20	0	5

b) **Pattern Parameters**

Pattern	Split	Seq.	Coord Mode	Det Plan
1	1	1	Auto Permissive ▼	1
2	2	2	Fixed Permissive ▼	2
3	3	3	Actuated Coord ▼	1
4	4	4	Single Permissive ▼	2

Figure 6.5. a) A day plan in MAXTIME. b) Patterns in MAXTIME

These forms allow traffic engineers to specify/update the timing configurations and monitor the intersections. For example, Figure 6.4 (a) shows the form used to communicate the status of an intersection.

Intelight Controller's Forms and Operation

Each intersection has a defined *day plan* for each day of the year. This plan reflects the controller's operation mode (i.e., *pre-timed*, *semi-actuated* or *fully actuated*) for the intersection. The form shown in Figure 6.5 (a) shows the specification of a day plan in terms of the *actions* that have to be executed throughout a day. An action is defined in terms of a *pattern* which specifies the timing configurations, i.e., operation mode, split plan, detector plan, sequence of phases, etc. (see Figure 6.4 (b) and Figure 6.5 (b)).

An Intelight controller selects the current day plan by accessing a *time-based schedule table* (see Figure 6.6 (a)) filled by traffic engineers. Each row, i.e., *schedule*, reflects a rule that indicates which plan is to be executed on specific days of a month. For example, in Figure 6.6 (a), traffic engineers have set schedule 3 to indicate that from August 15th to the 31st, on Mondays through Fridays, Plan 5 is to be executed. A specific day may be referenced in more than one rule (e.g., August 15th is included in schedules 1 and 3). In this

a)

Schedule	Month											Day of week							Day of month																							Day Plan
	J	F	M	A	M	J	J	A	S	O	N	D	S	M	T	W	T	F	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
1	J	F	M	A	M	J	J	A	S	O	N	D		M	T	W	T			1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	1
2					M									M	T	W	T	F				3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0											6
3								A						M	T	W	T	F																5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	5

b)

Schedule	Month											Day of week							Day of month																							Day Plan
	J	F	M	A	M	J	J	A	S	O	N	D	S	M	T	W	T	F	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
90										O				M																												2									20

Figure 6.6. a) The schedule table in MAXTIME. b) Schedule rule for DALI

a) **Input Points**

IO Module: 3

Input Point	Input Control Type	Index
1	Vehicle Det Call	1
2	Vehicle Det Call	2
3	Vehicle Det Call	3
4	Vehicle Det Call	4
5	Vehicle Det Call	5
6	Vehicle Det Call	6
7	Vehicle Det Call	7
8	Vehicle Det Call	8

b) **Input Points**

IO Module: 7

Input Point	Input Control Type	Index
1	Vehicle Det Call	33
2	Vehicle Det Call	34
3	Vehicle Det Call	35
4	Vehicle Det Call	36
5	Vehicle Det Call	37
6	Vehicle Det Call	38
7	Vehicle Det Call	39
8	Vehicle Det Call	40

Figure 6.7. a) Assignments of input points in MAXTIME. b) Assignments of virtual detectors for DALI.

case, the schedule defined for the least number of days in the calendar year (e.g., schedule 3) is selected. When the number of days is equal, the schedule with the smallest ID is selected.

Intelight Controller's Inputs and Outputs

The controller has I/O modules each with input and output points. I/O points can be either *active* or *inactive*. At the time of installation, traffic engineers connect each detector to a controller input point (see Figure 6.7 (a)). Figure 6.8 shows the forms that respectively display the status of input points and the status of detectors. Active detectors are marked with an "x".

a)

Input Points Status

IO Module: 1

Input Points	1	2	3	4	5	6	7	8
Status			●				●	
Control	☐	☐	☐	☐	☐	☐	☐	☐

b)

Vehicle Detector Status

Group	1	2	3	4	5	6	7	8
Active			X				X	
Alarms								

Figure 6.8. a) Status of input points in MAXTIME b) Status of detectors in MAXTIME.

a) **Day Plan Events**

Event	Hour	Minute	Action
19	15	43	128
20	15	45	1

b) **Pattern Parameters**

Pattern	Split	Seq.	Coord Mode	Det Plan
1	1	128	Auto Permissive	3

Figure 6.9. a) Dayplan for DALI. b) Pattern for DALI

6.5 Agent-Controller Interaction Mechanism

In order for a DALI agent to instruct a controller to perform an action, it was necessary to:

a) Define a new *pattern* for the agents. As mentioned above, a pattern specifies the timing constraints, i.e., maximum and minimum green, fully actuated mode, etc. (see Figure 6.9 (b)). The DALI pattern is associated with an *action* which is included in a *day plan*. Agents update the day plan every one minute (see Figure 6.9 (a)).

b) Define *virtual detectors*. These *virtual detectors* are used by the agents to simulate a detector state (*active* or *inactive*) and therefore trigger the execution of a signal change by the controller. In DALI mode, a controller operates in fully actuated mode. As such, it always gives green to a phase when a vehicle is detected for that phase and no vehicles are detected for the other phases. When an agent wants a signal to be green, it changes the status of the virtual detector to *active* and all the other virtual detectors to *inactive* (see Figure 6.8 (a)). This prompts the controller to give green to that phase. The same

approach is followed for a combination of non-conflicting phases. From an implementation perspective, the virtual detectors controlled by the agents are connected to a spare Intelight controller's input point (see Figure 6.7(b)).

6.6 Executing DALI Agents

At initialization time, a DALI agent connects to its controller's web server through VPN, adds a new rule to the controller's schedule table (see Figure 6.6 (b)) to execute the DALI day plan and takes control.

It then requests an update on the status of the intersection. The Intelight controller updates the phase status form (see Figure 6.4 (a) which shows the state of its traffic lights (i.e., red, green or yellow) and detectors (i.e., active or inactive). This information is read by the agent. The time interval between the agent request and the controller update called *communication speed* has to be less than three hundred milliseconds. The agent then proceeds by executing the steps discussed in Section 2.3.

To release control to the Intelight controller, an agent only needs to remove the rule from the schedule table.

6.7 Lessons Learned

DALI was thoroughly verified by traffic engineers and intensively tested through simulation (Torabi et al., 2017, 2018a,c,b). The first execution of DALI ran smoothly, and agents behaved as specified. The excitement related to a glitch-free deployment soon dissipated when we realized that several real-world scenarios were not considered in the problem definition.

Loss of Communication. When an agent loses communication with its controller, the virtual detector that was given the status *active* by the agent remains in that status for as long as the communication is interrupted. This results in giving maximum green to the phase associated with the detector, and minimum green to all other phases. In order to

avoid this situation, we decided that if an agent is disconnected from its controller for over two minutes, the controller is given control of the intersection.

Time Synchronization. In the simulated traffic environment, we assumed that all agents use the same clock time. After deployment, we realized that the controller's clock values were not synchronized. As a result, when an agent sends a vehicle detection time to its neighbor, the neighbor computes the estimated arrival time for that vehicle incorrectly. To address this issue, we added a *time-server* to DALI. When an agent detects a vehicle, it sends the detection time using the *time-server* time and not the controller time.

Detector Failures. We realized that when a detector fails, two things happen: the detector status (i.e., *active*, *inactive*) stays the same for a long period of time, or it changes randomly. We also realized that the Intelight controllers do not have a mechanism to address detector failures. We improved our agent algorithms to consider this case as follows: when an agent notices that a detector has failed, it replaces the actual detector data with an estimate of the data. This estimate is computed using prediction models and historical data.

Road Construction. While DALI was running, some road construction work was done at the Franklyn intersection. During construction, the topology of the intersection was changed. This required a re-definition of the intersection structure for the agent controlling the Franklyn intersection.

Rain: During heavy rainy days, the communication speed between agents and controllers decreased drastically. In addition, detectors did not work properly (i.e., they would detect some vehicles but not all). Given that an agent's performance highly depends on the communication with its controller and the detection of vehicles, we decided that when the communication speed is below 300ms, an agent has to give control to the Intelight controllers.

CHAPTER 7

EVALUATION OF DALI'S DEPLOYMENT RESULTS

7.1 Metrics and Data Collection

Various metrics are commonly used in Texas to evaluate the performance of a signal timing system. We evaluate DALI with respect to *queue length*, *delay*, and *cost of the delay*. Queue length is defined as the number of vehicles stopped at the intersection and is computed as discussed in Section 2.3. As mentioned in Section 5.1, delay is the increment in a vehicle's travel time caused by traffic control devices, compared with the travel time if the vehicle was to maintain its expected speed in the absence of any control device. The cost of delay is computed using data from (Cookson and Pishue, 2017).

The results presented in this chapter are based on data collected over a period of 520 hours as follows: we ran DALI and gathered data for 260 hours. Then we turned off DALI and gathered data for SCATS-R for the remaining 260 hours. An initial analysis of traffic on Waterview showed that overall, traffic variations occur between 7:00 to 9:00 and 16:00 to 20:00; 9:00 to 16:00 and 20:00 to 00:00; and 00:00 to 7:00 (see Figure 7.1).

We considered week days and weekends, and given the traffic variations discussed above, divided days into three time periods: peak hours (i.e., 7:00 to 9:00 and 16:00 to 20:00), off-peak hours (i.e., 9:00 to 16:00 and 20:00 to 00:00), and nighttime (i.e., 00:00 to 7:00). Table 7.1 shows the distribution of the 520 data collection hours.

7.2 Queue Length

We first analyze the data with respect to reduction in queue lengths at each intersection. As shown in Figure 7.2, for all phases, DALI agents reduce queue lengths at all intersections. In average, for all three intersections, queue length is reduced by 39.43% (49.43% for SY, 38.32% for FER and 30.53% for FO).

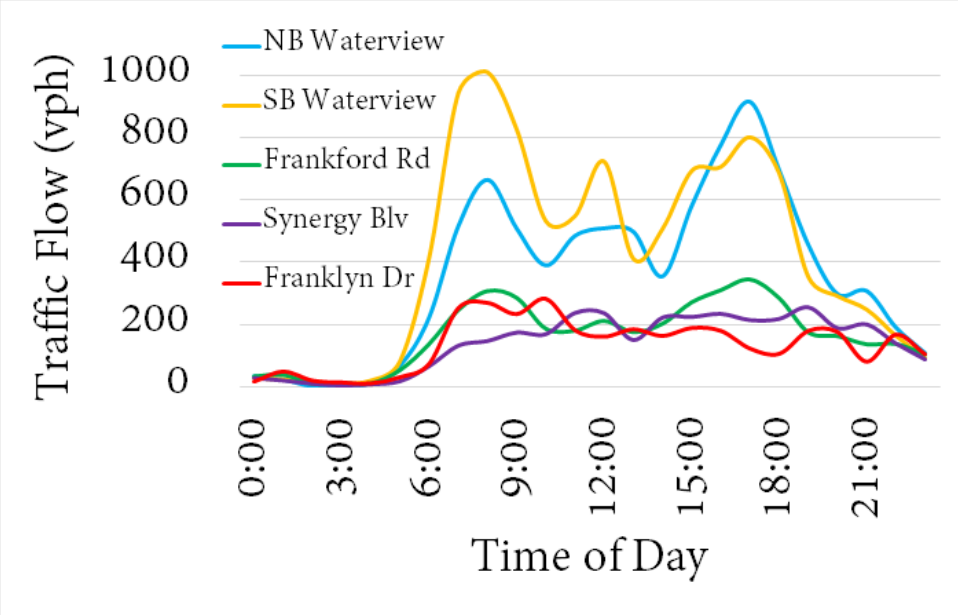

Figure 7.1. Traffic Flow Rate for Different Times of the Day.

Table 7.1. Distribution of data collection time

Dali - Duration of Control (Hours)				
	Peak	Off-peak	Midnight	Total
Weekdays	38.49	67.37	36.16	142.02
Weekends	29.84	53.87	34.25	117.61
Total	67.97	121.24	70.42	259.63
SCATS-R - Duration of Control (Hours)				
	Peak	Off-peak	Midnight	Total
Weekdays	54.65	93.09	40.64	189.20
Weekends	15.08	29.83	25.68	70.59
Total	69.73	123.74	66.32	259.80

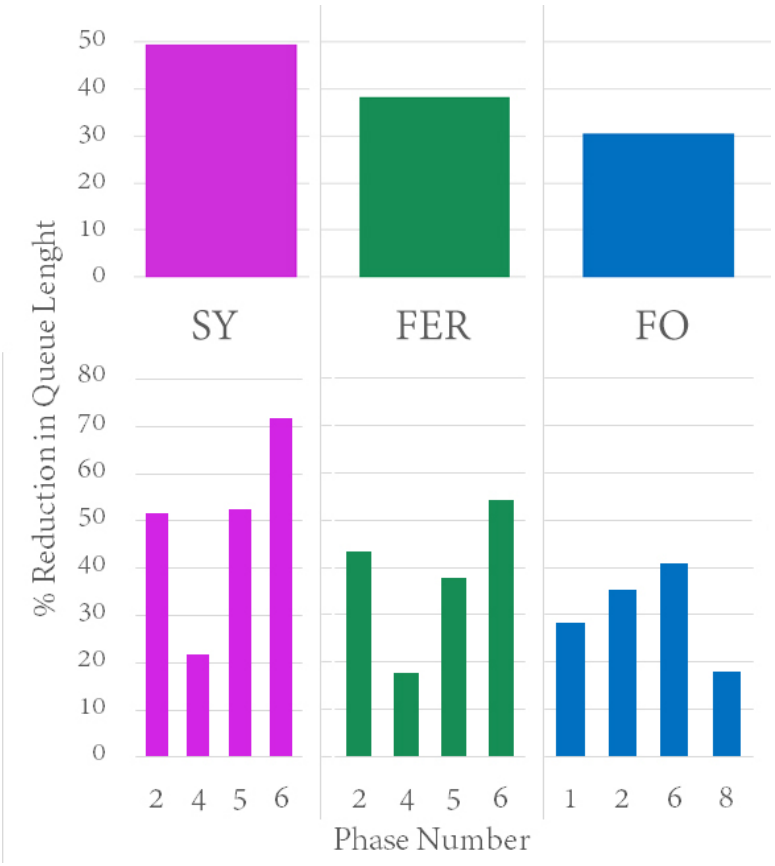

Figure 7.2. Queue Length Reduction.

This reduction is higher for phases 6, 2 and 5 on SY and FER, and phases 6, 2 and 1 on FO. It is lower for phases 8 and 4. This is due to the fact that information about vehicle arrivals is not available for these phases either because the upstream intersections are not under the control of the City of Richardson (case of FO), or the upstream intersections do not provide detector information (case of SY and FER). Figure 7.2 shows that for SY, the reduction in queue length for Phase 6 is higher than for Phase 2 although for both phases, SY's agent receives information about vehicle arrivals. This is due to the fact that the distance between SY and FER is twice the distance between SY and FO. Therefore, SY's agent has more flexibility in scheduling green signals for Phase 6 given that it receives information about vehicle arrivals for Phase 6 earlier than for Phase 2.

7.3 Delay

Figure 7.3 shows the average reduction in delay for different time periods. We notice that there is a high reduction in delay for time periods *week day nighttime*, *weekend peak & off peak* and *weekend nighttime*. This is explained by the fact that, during nighttime (on weekdays and weekends), traffic is very slow and there are only very few requests for green signals at the intersections. In SCATS-R mode, once a vehicle is detected, the Intelight controllers either give green immediately or, if in the middle of a cycle, complete the sequence then give green. In DALI mode, given that the agents can predict the arrival of vehicles, they schedule green signals before the vehicles arrive at the intersection. On weekends, during the day (peak and off-peak), although traffic is slightly heavier than during nighttime, the same behavior occurs.

Figure 7.3 shows that the lowest reduction in delay happens for *weekday off-peak* hours. The 12.81% difference between *weekday peak* hours and *weekday off-peak* hours is due to the fact that during peak hours, intersections are likely to be congested and the agent collaboration process is superior to the SCATS-R timing strategy. During off-peak hours, the number of vehicles although high does not necessarily lead to congestion. Therefore agents are unlikely to collaborate and their performance is not as superior.

7.4 Cost of Delay

Figure 7.4 shows the average delay (in seconds) and the corresponding average cost of delay (in US $) for each vehicle at each intersection for both SCATS-R and DALI. According to (Cookson and Pishue, 2017), in the Dallas area, the cost of delay for a driver is on average $31 per hour which translate into $0.0086 per second. For *weekdays peak* hours, in SCATS-R mode, the average delay is 43.2 seconds. In the same period, in DALI mode, the average delay is 24.38 seconds and therefore delay reduction is on average $43.2 - 24.38 = 18.8$ seconds

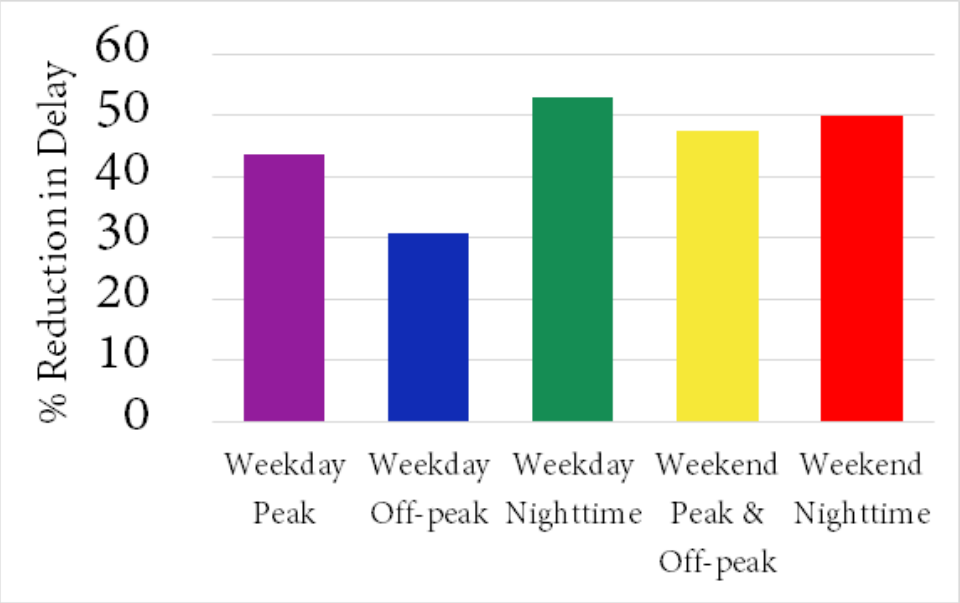

Figure 7.3. Delay Reduction.

which translates into a savings of $18.8 \times \$0.0086 = \$0.16$ per vehicle per intersection. During that period, an average of $1,604$ vehicles/hour passed through a Waterview intersection. Therefore the average savings for all the vehicles going through an intersection is $\$1,604 \times \$0.16 = \$256.64$/hour.

We computed the average delay for SCATS-R and DALI during their respective 260 hours of execution time. This was achieved by computing the difference between the arrival time and the departure time of each vehicle at each intersection. The value of the average delay for a vehicle at an intersection for SCATS-R is 27.94 seconds whereas for DALI it is 16.73 seconds. Therefore delay reduction is on average $27.94 - 16.73 = 11.21$ seconds which translates into a savings of $11.21 \times \$0.0086 = \$0.1$ per vehicle per intersection. During the 260 hours of execution, an average of 725 vehicles/hour pass through a Waterview Pkwy intersection. Therefore the average savings for vehicles going through that intersection is $725 \times \$0.1 \times 260 = \$18,850$ overall.

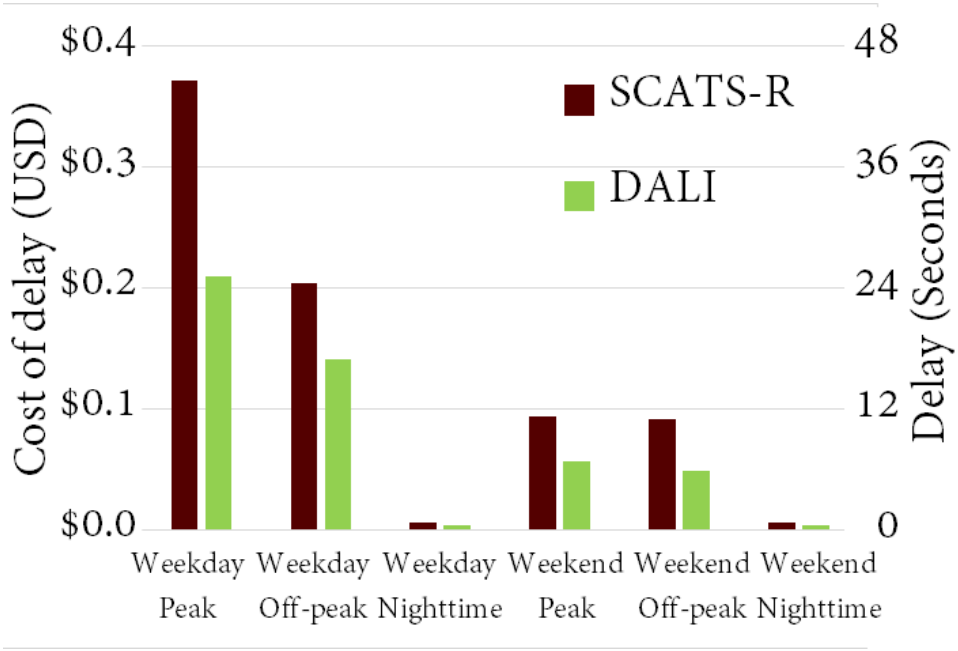

Figure 7.4. Average delay and cost of the delay for each vehicle at each intersection.

Our initial analysis indicates that a deployment of DALI agents on the City of Richardson's 128 signalized intersections may reduce the cost of the delay for the city's drivers by tens of million dollars every year.

CONCLUSION

In this work we presented DALI, a collaborative multi-agent traffic signal system and its deployment in the City of Richardson's Waterview Parkway corridor at three major intersections.

7.5 Contributions

Among our contributions we mention the following:

7.5.1 Model and Algorithm Definitions

- We defined and implemented algorithms for DALI, a multi-agent coordinated traffic signal timing system.
- We defined and implemented algorithms for adaptive assignment of threshold values.
- We tested DALI through a) simulation using real-world traffic data provided by the City of Richardson, and b) hybrid simulation using real-time traffic data.

7.5.2 Traffic Simulator Feature Development

We extended the MATISSE agent-based traffic simulation system to include several new features. These include:

- A tool to automatically convert OpenStreetMap's traffic network maps into MATISSE's format.
- A feature which allows the simulation of real-world traffic detection systems, e.g., inductive loops.
- A feature which allows the simulation of real-world intersection controllers' operation modes, e.g., pre-timed, semi-actuated.

- A tool which reads a database of stored real-world traffic data and populates a MATISSE simulation with this data.

- A feature which allows the integration of MATISSE with MaxTime.

- A feature that allows MATISSE to process real-time traffic data provided by MaxTime and run *hybrid* simulations.

- A feature that allows simulated controller agents to send commands to deployed Intelight controllers through MaxTime.

7.5.3 Deployment

- We developed DALI controller agents.

- In preparation for deployment, we created a virtual Intelight controller in the lab to test its integration with DALI agents.

- We developed a tool to monitor the operations of DALI's agents.

- We investigated possible approaches to safely integrate DALI agents and Intelight controllers in collaboration with the City of Richardson.

- We successfully ran DALI controller agents on three intersections in the Waterview Parkway corridor.

7.6 Future Work

Future work includes the investigation of the following open problems:

7.6.1 Model and Algorithm Definitions

- In the current DALI model, predictions (e.g., vehicle arrival, vehicle movement) are made by the agents based on historical data. We plan to investigate the use of simulation by the agents for prediction purposes.

- During the decision making step and in certain situations, the agents generate a very large number of plans. This number can be reduced.

7.6.2 Traffic Simulator Feature Development

- In the current version of MATISSE, it is not possible to simulate traffic networks for countries with left-hand traffic. We plan to add this feature.

- Due to the limitation in the communication bandwidth, it is not possible to integrate more than a few intersection controllers in the hybrid simulation. We plan to investigate new techniques to increase the number of controllers in the hybrid simulation.

7.6.3 Deployment

- The algorithm that agents use to identify failed detectors does not work properly in some situations and needs to be improved.

- When an agent loses its connection with the controller, it does not sends information about vehicle arrivals to its neighbors. We plan to add a feature to enable agents to send this information to their neighbors based on historical data.

- During road constructions, we had to manually update the network topology for the agents. We plan to add a feature to enable agents to automatically detect changes in the topology of the road network.

- In the current system architecture, agents run on computers and send commands to intersection controllers through VPN. We plan to investigate executing agents in the intersection controller boxes and comparing the performance of these two architectures.
- We plan to extend the deployment of DALI to critical intersections in the City of Richardson.
- We plan to connect vehicles to DALI to allow for vehicle-to-infrastructure interactions.

REFERENCES

Abdulhai, B., R. Pringle, and G. J. Karakoulas (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering 129*(3), 278–285.

Al-Zinati, M. and R. Zalila-Wenkstern (2015). Matisse 2.0: A large-scale multi-agent simulation system for agent-based its. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on*, Volume 2, pp. 328–335. IEEE.

Araghi, S., A. Khosravi, and D. Creighton (2015). A review on computational intelligence methods for controlling traffic signal timing. *Expert systems with applications 42*(3), 1538–1550.

Balke, K. N. and C. Herrick (2004, July). Potential measures of assessing signal timing performance using existing technologies. Technical Report FHWA/TX-04/0-4422-1, Texas A&M Transportation Institute, College Station, Texas 77843-3135.

Bazzan, A. L. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems 10*(1), 131–164.

Bazzan, A. L. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems 18*(3), 342–375.

Bazzan, A. L., D. de Oliveira, and B. C. da Silva (2010). Learning in groups of traffic signals. *Engineering Applications of Artificial Intelligence 23*(4), 560–568.

Bazzan, A. L. and F. Klügl (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review 29*(3), 375–403.

Bi, Y., D. Srinivasan, X. Lu, Z. Sun, and W. Zeng (2014). Type-2 fuzzy multi-intersection traffic signal control with differential evolution optimization. *Expert Systems with Applications 41*(16), 7338–7349.

Bui, K.-H. N., J. E. Jung, and D. Camacho (2017). Game theoretic approach on real-time decision making for iot-based traffic light control. *Concurrency and Computation: Practice and Experience 29*(11), e4077.

Chao, K.-H., R.-H. Lee, and M.-H. Wang (2008). An intelligent traffic light control based on extension neural network. In *Knowledge-based intelligent information and engineering systems*, pp. 17–24. Springer.

Chen, B. and H. H. Cheng (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems 11*(2), 485–497.

Chen, O. and M. Ben-Akiva (1998). Game-theoretic formulations of interaction between dynamic traffic control and dynamic traffic assignment. *Transportation Research Record: Journal of the Transportation Research Board* (1617), 179–188.

Cheng, S.-F., M. A. Epelman, and R. L. Smith (2006). Cosign: A parallel algorithm for coordinated traffic signal control. *IEEE Transactions on Intelligent Transportation Systems 7*(4), 551–564.

Chin, Y. K., L. K. Lee, N. Bolong, S. S. Yang, and K. T. K. Teo (2011). Exploring q-learning optimization in traffic signal timing plan management. In *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 269–274. IEEE.

Collotta, M., L. L. Bello, and G. Pau (2015). A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers. *Expert Systems with Applications 42*(13), 5403–5415.

Cookson, G. and B. Pishue (2017). Inrix global traffic scorecard. *INRIX Research, February*.

Day, C. M., T. M. Brennan Jr, H. Premachandra, A. M. Hainen, S. M. Remias, J. R. Sturdevant, G. Richards, J. S. Wasson, and D. M. Bullock (2010, October). Quantifying benefits of traffic signal retiming. Final Report FHWA/IN/JTRP-2010/22, Joint Transportation Research Program, Purdue University, 610 Purdue Mall, West Lafayette, IN 47907.

De Oliveira, D., A. L. Bazzan, and V. Lesser (2005). Using cooperative mediation to coordinate traffic lights: a case study. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 463–470. ACM.

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

Dong, C., Z. Liu, and Z. Qiu (2005). Urban traffic signal timing optimization based on multilayer chaos neural networks involving feedback. In *International Conference on Natural Computation*, pp. 340–344. Springer.

Dresner, K. and P. Stone (2008). A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research 31*, 591–656.

Dusparic, I. and V. Cahill (2012). Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Transactions on Autonomous and Adaptive Systems (TAAS) 7*(1), 11.

Dusparic, I., J. Monteil, and V. Cahill (2016). Towards autonomic urban traffic control with collaborative multi-policy reinforcement learning. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 2065–2070. IEEE.

El-Tantawy, S. and B. Abdulhai (2010). An agent-based learning towards decentralized and coordinated traffic signal control. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 665–670. IEEE.

El-Tantawy, S., B. Abdulhai, and H. Abdelgawad (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems 14*(3), 1140–1150.

Elhenawy, M., A. A. Elbery, A. A. Hassan, and H. A. Rakha (2015). An intersection game-theory-based traffic control algorithm in a connected vehicle environment. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 343–347. IEEE.

France, J. and A. A. Ghorbani (2003). A multiagent system for optimizing urban traffic. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pp. 411–414. IEEE.

Gartner, N. H. (1982). Development and testing of a demand-responsive strategy for traffic signal control. In *1982 American Control Conference*, pp. 578–583. IEEE.

Gartner, N. H., F. J. Pooran, and C. M. Andrews (2001). Implementation of the opac adaptive control strategy in a traffic signal network. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 195–200. IEEE.

Ghanim, M. S. and G. Abu-Lebdeh (2015). Real-time dynamic transit signal priority optimization for coordinated traffic networks using genetic algorithms and artificial neural networks. *Journal of Intelligent Transportation Systems 19*(4), 327–338.

Henry, J.-J., J. L. Farges, and J. Tuffal (1983). The prodyn real time traffic algorithm. *IFAC Proceedings Volumes 16*(4), 305–310.

Hunt, P., D. Robertson, R. Bretherton, and M. C. Royle (1982, April). The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control 23*(4), 190–192.

Intelight (2019). https://www.intelight-its.com/. Accessed February 2019.

Koonce, P., L. Rodegerdts, K. Lee, S. Quayle, S. Beaird, C. Braud, J. Bonneson, P. Tarnoff, and T. Urbanik (2008, June). Traffic signal timing manual. Publication FHWA-HOP-08-024, Department of Transportation, 1200 New Jersey Ave, SE, Washington, DC 20590.

Kosonen, I. (2003). Multi-agent fuzzy signal control based on real-time simulation. *Transportation Research Part C: Emerging Technologies 11*(5), 389–403.

Li, L., Y. Lv, and F.-Y. Wang (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica 3*(3), 247–254.

Li, L., D. Wen, and D. Yao (2014). A survey of traffic control with vehicular communications. *IEEE Transactions on Intelligent Transportation Systems 15*(1), 425–432.

Liu, Z. (2007). A survey of intelligence methods in urban traffic signal control. *IJCSNS International Journal of Computer Science and Network Security 7*(7), 105–112.

Lu, S., X. Liu, and S. Dai (2008). Incremental multistep q-learning for adaptive traffic signal control based on delay minimization strategy. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pp. 2854–2858. IEEE.

Mannion, P., J. Duggan, and E. Howley (2015a). Parallel reinforcement learning for traffic signal control. Volume 52, pp. 956–961. Elsevier.

Mannion, P., J. Duggan, and E. Howley (2015b). Parallel reinforcement learning for traffic signal control. *Procedia Computer Science 52*, 956–961.

Mannion, P., J. Duggan, and E. Howley (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pp. 47–66. Springer.

Mashayekhi, M. and G. List (2015). A multi-agent auction-based approach for modeling of signalized intersections. In *Second Workshop on Synergies Between Multiagent Systems, Machine Learning and Complex Systems (TRI 2015), Buenos Aires, Argentina*.

Mirchandani, P. and L. Head (2001, December). A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies 9*(6), 415–432.

Papageorgiou, M., C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang (2003). Review of road traffic control strategies. *Proceedings of the IEEE 91*(12), 2043–2067.

PTV-Group (2018). Ptv vissim. http://vision-traffic.ptvgroup.com/en-us/products/ptvvissim/. Accessed October 2018.

Richter, S. (2006). Learning traffic control-towards practical traffic control using policy gradients. *Albert-Ludwigs-Universitat Freiburg, Tech. Rep*.

Roozemond, D. A. (2001). Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research 131*(2), 293–301.

Saito, M. and J. Fan (2000). Artificial neural network–based heuristic optimal traffic signal timing. *Computer-Aided Civil and Infrastructure Engineering 15*(4), 293–307.

SCATS, T. R. . M. S. (2019). Scats: The benchmark in urban traffic control. http://www.scats.com.au/. Accessed February 2019.

SCOOT, U. T. R. L. (2019). Split cycle and offset optimisation technique. https://trlsoftware.co.uk/products/traffic_control/scoot. Accessed February 2019.

Sims, A. G. and K. W. Dobinson (1980). The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology 29*(2), 130–137.

Srinivasan, D., M. C. Choy, and R. L. Cheu (2006). Neural networks for real-time traffic signal control. *IEEE Transactions on Intelligent Transportation Systems 7*(3), 261–272.

Sun, D., R. F. Benekohal, and S. T. Waller (2006). Bi-level programming formulation and heuristic solution approach for dynamic traffic signal optimization. *Computer-Aided Civil and Infrastructure Engineering 21*(5), 321–333.

Teodorović, D., V. Varadarajan, J. Popović, M. R. Chinnaswamy, and S. Ramaraj (2006). Dynamic programming neural network real-time traffic adaptive signal control algorithm. *Annals of Operations Research 143*(1), 123–131.

Torabi, B., M. Al-Zinati, and R. Z. Wenkstern (2018). Matisse 3.0: A large-scale multi-agent simulation system for intelligent transportation systems. In *Proceedings of the 16th International Conference on Practical Applications of Agents and Multi-Agent Systems*, PAAMS 18, Toledo, Spain, pp. 357–360.

Torabi, B., R. Z. Wenkstern, and M. Al-Zinati (2018). An agent-based micro-simulator for its. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2556–2561. IEEE.

Torabi, B., R. Z. Wenkstern, and R. Saylor (2017, October). Agent-based decentralized traffic signal timing. In *Proceedings of the 21st International Symposium on Distributed Simulation and Real Time Applications*, DS-RT 17, Rome, Italy, pp. 123–126.

Torabi, B., R. Z. Wenkstern, and R. Saylor (2018a, November). A collaborative agent-based traffic signal system for highly dynamic traffic conditions. In *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE ITSC 2018, Maui, Hawaii, USA, pp. 626–633.

Torabi, B., R. Z. Wenkstern, and R. Saylor (2018b, July). A multi-hop agent-based traffic signal timing system for the city of richardson. In *Proceedings of the The Sixteenth International Conference on Autonomous Agent and Multiagent Systems*, AAMAS 2018, Stockholm, Sweden, pp. 2094–2096.

Torabi, B., R. Z. Wenkstern, and R. Saylor (2018c, September). A self-adaptive collaborative multi-agent based traffic signal timing system. In *Proceedings of the 4th IEEE International Smart Cities Conference*, ISC2 2018, Kansas City, Missouri, USA.

Trafitek (2019). Utopia. http://www.trafitek.com/atc-utopia.php. Accessed February 2019.

TRANSYT, U. T. R. L. (2019). Traffic network and isolated intersection study tool. https://trlsoftware.co.uk/products/junction_signal_design/transyt. Accessed February 2019.

Wen, K., S. Qu, and Y. Zhang (2007). A stochastic adaptive control model for isolated intersections. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pp. 2256–2260. IEEE.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Yau, K.-L. A., J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk (2017). A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys (CSUR) 50*(3), 34.

Younes, M. B. and A. Boukerche (2016). Intelligent traffic light controlling algorithms using vehicular networks. *IEEE transactions on vehicular technology 65*(8), 5887–5899.

Zhao, Y. and Z. Tian (2012). An overview of the usage of adaptive signal control system in the united states of america. In *Applied Mechanics and Materials*, Volume 178, pp. 2591–2598. Trans Tech Publ.

Zhen-long, L. (2003). A differential game modeling approach to dynamic traffic assignment and traffic signal control. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, Volume 1, pp. 849–855. IEEE.

Zohdy, I. H. and H. Rakha (2012). Game theory algorithm for intersection-based cooperative adaptive cruise control (cacc) systems. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 1097–1102. IEEE.

BIOGRAPHICAL SKETCH

Behnam Torabi was born in Isfahan, Iran in 1986. After completing his schoolwork at Alborz High School in Isfahan in 2003, Behnam entered the University of Isfahan in which he earned a Bachelor of Science with a major in Software Engineering and a Master of Science in Artificial Intelligence. In January 2015, he moved to Dallas, TX and entered The University of Texas at Dallas. He was awarded a Master of Science in Software Engineering from The University of Texas at Dallas in 2017.

CURRICULUM VITAE

Behnam Torabi

Address: ECSS 4.220, 800 W Campbell Rd, Richardson, TX 75080

Email: behnam.torabi@utdllas.edu

Education

2017 - M.Sc. in Software Engineering, University of Texas at Dallas

2013 - M.Sc. in Artificial intelligence, University of Isfahan, Iran

2010 - B.Sc. in Software Engineering, University of Isfahan, Iran

Publications

Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. A Self-Adaptive Collaborative Multi-Agent based Traffic Signal Timing System. In Proceedings of the 4th IEEE International Smart Cities Conference, ISC2 2018.

Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. A Collaborative Agent-Based Traffic Signal System for Highly Dynamic Traffic Conditions. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, IEEE ITSC 2018.

Behnam Torabi, Rym Z. Wenkstern, and Mohammad Al-Zinati. An Agent-Based Micro-Simulator for ITS. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, IEEE ITSC 2018, Maui, Hawaii, USA, November 2018.

Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. A Multi-Hop Agent-Based Traffic Signal Timing System for the City of Richardson. In Proceedings of the Sixteenth International Conference on Autonomous Agent and Multiagent Systems, AAMAS 2018.

Behnam Torabi, Mohammad Al-Zinati, and Rym Z. Wenkstern. MATISSE 3.0: A Large-Scale Multi-Agent Simulation System for Intelligent Transportation Systems. In Proceedings of the 16th International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAMS 18.

Behnam Torabi, Rym Z. Wenkstern, and Robert Saylor. Agent-based decentralized traffic signal timing. In Proceedings of the 21st International Symposium on Distributed Simulation and Real Time Applications, DS-RT 17.

Behnam Torabi and Ahmad Reza Naghsh Nilchi. Automatic Speech Segmentation Based on Audio and Optical Flow Visual Classification. In International Journal of Image, Graphics and Signal Processing.

Awards & News

April 2019 - Winner of Smart 50 Awards
April 2019 - Interview with Smart Cities Connect
March 2019 - News Segment in CBS 11
February 2019 - Article in Dallas Innovates

Technical Skills

JAVA, C#, C/C++, Php, ASP.NET, HTML5, JavaScript, AJAX, CSS, MATLAB, LaTeX, Software Engineering, Artificial Intelligence, Software Design, Signal Processing, Image Processing