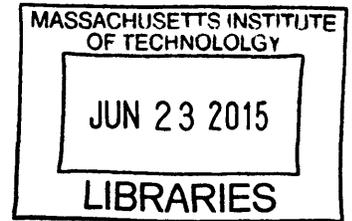


**High Performance and Provably Safe  
Polling-Systems-Based Control Algorithms of  
All-Autonomous Intersections**

**ARCHIVES**



by

David Miculescu

Submitted to the Department of Aeronautics and Astronautics  
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.

**Signature redacted**

Author .....

Department of Aeronautics and Astronautics

May 21, 2015

**Signature redacted**

Certified by .....

Sertac Karaman

Associate Professor

Thesis Supervisor

**Signature redacted**

Accepted by .....

Paulo C. Lozano

Associate Professor of Aeronautics and Astronautics

Chair, Graduate Program Committee



# **High Performance and Provably Safe Polling-Systems-Based Control Algorithms of All-Autonomous Intersections**

by

David Miculescu

Submitted to the Department of Aeronautics and Astronautics  
on May 21, 2015, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Aeronautics and Astronautics

## **Abstract**

The rapid development of autonomous vehicles spurred a careful investigation of the potential benefits of all-autonomous transportation networks. Most studies conclude that autonomous systems can enable drastic improvements in performance. A widely studied concept is all-autonomous, collision-free intersections, where vehicles arriving in a traffic intersection with no traffic light adjust their speeds to cross through the intersection as quickly as possible. In this thesis, we propose a coordination control algorithm for this problem, assuming stochastic models for the arrival times of the vehicles. The proposed algorithm provides provable guarantees on safety and performance. More precisely, it is shown that no collisions occur surely, and moreover a rigorous upper bound is provided for the expected wait time. The algorithm is also demonstrated in simulations. The proposed algorithms are inspired by polling systems. In fact, the problem studied in this thesis leads to a new polling system where customers are subject to differential constraints, which may be interesting in its own right.

Thesis Supervisor: Sertac Karaman  
Title: Associate Professor



## **Acknowledgments**

I wish to acknowledge first and foremost my Creator who endows me with the inalienable gift of life anew each morning, with an eternal purpose in my work, and constant creativity in my research.

Secondly, I wish to acknowledge my advisor Dr. Sertac Karaman, whose inspiration, insights, guidance and instruction has been instrumental in this work. I sincerely appreciate his devotion and patience throughout the entirety of this work.

Thirdly, I wish to acknowledge my labmate and friend Fangchang Ma for giving me meaningful feedback regarding my research over the past years. Moreover, thank you for helping create an exciting lab atmosphere, one in which working was a joy and not a duty.

Fourthly, I would like to acknowledge my community here in Boston for their constant support, thoughtfulness, and prayers over these last couple of years. In particular, my Sidney Pacific Small Group lead by Peng Shi and Annie Chen, my roommate and brother Sebastian Dumulesc for always believing in me, my sisters Felicia Ciuculescu and Simina Petrovan for sustenance when time became a scarce resource, as well as the rest of the members of my Romanian community “Sfintii din Boston”.

Last but not least, I would like to acknowledge my parents for their constant presence in my life. Without their support, advice, knowledge, care, and love throughout the years, I would not be here in Boston, studying at one of the world’s finest institutions. I deeply thank you all.



# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Motivation . . . . .	13
1.2	Related Work on All-Autonomous Intersections . . . . .	15
1.3	Thesis Contributions . . . . .	19
<b>2</b>	<b>Problem Definition</b>	<b>21</b>
<b>3</b>	<b>Control Policy</b>	<b>25</b>
3.1	Background on Queueing and Polling Systems . . . . .	25
3.2	Simulating Polling Systems Behavior . . . . .	28
3.3	The Intersection Coordination Algorithm . . . . .	29
<b>4</b>	<b>Analysis</b>	<b>33</b>
4.1	Arrival time model . . . . .	33
4.2	Assumptions . . . . .	34
4.3	Main theoretical results . . . . .	36
<b>5</b>	<b>Computational Experiments</b>	<b>53</b>
5.1	Approximate Vehicle Trajectories . . . . .	53
5.2	Overcrowding Assumption . . . . .	57
5.3	Traffic Light Comparison . . . . .	59
<b>6</b>	<b>Conclusion</b>	<b>63</b>



# List of Figures

1-1	A warehouse populated with Kiva Systems LLC robots, automating warehouse logistics [10]. . . . .	13
1-2	Unmanned automated guided vehicles (AGV) transporting containers in Europe’s largest sea port, in Rotterdam, Netherlands [30]. . . . .	14
1-3	Illustration of an autonomous vehicle approaching an intersection with an uncooperative manual driver [24]. . . . .	16
1-4	Illustration of an autonomous vehicle with a computationally efficient, hybrid controller that preserves safety at intersection [9]. . . . .	17
1-5	Two intersecting streams of aircraft [23]. . . . .	18
1-6	AIM algorithm for scheduling vehicles approaching an autonomous intersection [17]. . . . .	18
2-1	An illustration of a fully autonomous traffic intersection. Autonomous vehicles arriving near the intersection are fully controlled by a central control system. The control system ensures that the vehicles safely pass through the intersection; furthermore, the central control system provides provable guarantees on performance, <i>e.g.</i> , an upper bound on the average time it takes a typical vehicle to go through this intersection region. . . . .	22
2-2	An illustration of the control region. . . . .	23
2-3	Graphical interpretation of the delay of a vehicle as given by Definition 2.0.2.	24

3-1	We show the three time instances that are described in the text. In (a), the front bumper of the vehicle is entering the intersection region. In (b), the rear bumper is leaving the control region and entering the intersection region. In (c), the rear bumper is leaving the intersection region. In the figure above, the control region (shaded light grey) and the intersection region (shaded in dark grey) are shown. . . . .	31
4-1	Phase portrait for Case 1 where there is no intersection point C. The $x$ -axis represents distance $\xi$ from intersection, point $I$ . The $y$ -axis plots velocity. The dashed curve is $\partial F_j$ . . . . .	46
4-2	Phase portrait for Case 1 where there is an intersection point C. The $x$ -axis represents distance $\xi$ from intersection, point $I$ . The $y$ -axis plots velocity. The dashed curve is $\partial F_j$ . . . . .	46
5-1	The trajectories of the vehicles are shown. The top, middle, and bottom figures are light, medium, and heavy load cases, respectively. The top half of each figure shows the trajectories of vehicles in Lane 1, and the bottom half is the trajectories of vehicles from Lane 2. Each figure shows a small window of time after the system reaches steady state. . . . .	55
5-2	The trajectories of the vehicles are shown according to a $k$ -limited polling policy, where $k = 4$ in the upper plot and $k = 8$ in the lower plot. . . . .	56
5-3	The comparison of the performance of the proposed algorithm with the performance of the corresponding polling system. . . . .	57
5-4	Percentage of thinned vehicles as a function of arrival process intensity. The length of the control region is set to $L^* = 50$ meters. . . . .	58
5-5	Log of the intensity of thinned vehicles, in units of $\log(\text{veh}/\text{sec})$ , is plotted against control region length $L$ . . . . .	59
5-6	Proposed algorithm expected wait time is compared with a traditional traffic simulation with green and red light phases of 5 seconds, 10 seconds, and 15 seconds. . . . .	61

# List of Tables

4.1 Latest Schedule Time Trajectory . . . . .	43
-----------------------------------------------	----



# Chapter 1

## Introduction

### 1.1 Motivation

Autonomous systems technology holds the potential to provide substantial performance increases in transportation systems, independently of the application domain. For example, robotic vehicles servicing packaging requests in large Amazon warehouses already increase efficiency [13, 34]. Similar autonomous vehicles provide material handling services in seaports [1, 12, 16]. In both application domains, the vehicles coordinate their motion in order to share common resources, in this case roads and intersections, as effectively as possible, for example to minimize transportation times.

Beyond facilitating routine tasks, all-autonomous vehicle systems enable drastic improvements in efficiency in dangerous work environments. As we seek to improve the performance of these currently human-operated networks, such as transportation networks



Figure 1-1: A warehouse populated with Kiva Systems LLC robots, automating warehouse logistics [10].



Figure 1-2: Unmanned automated guided vehicles (AGV) transporting containers in Europe’s largest sea port, in Rotterdam, Netherlands [30].

and air traffic management, human operators will simply not be able to react reliably and quickly enough to guarantee safety in this high performance regime, necessitating the development and implementation of all-autonomous vehicle systems.

Furthermore, in the near future, one can imagine the airspace above urban environments being utilized for aerial transport systems. For example, Amazon Prime Air boasts a maximum delivery time of thirty minutes with their emerging unmanned aerial vehicle (UAV) delivery system [3]. In March of 2015 the Federal Aviation Administration in the U.S. issued an experimental airworthiness certificate to Amazon, allowing the testing of drones in US airspace under certain conditions [2]. Although many restrictions are currently imposed, some quite stringent, such as the drone remaining within “visual line-of-sight of the pilot and observer,” as the safety of this emerging system is verified and validated, the FAA will be more inclined to relax this initial mandate, giving more freedom to Amazon Prime Air and other projects such as Google’s Project Wing. With the possibility of certain zones of our future airspace being populated, even congested, with drones, it is important to develop algorithms that safely motion plan these drones, while also seeking to globally determine what is the best strategy of routing these vehicles to their destinations.

Encouraged by the growing interest in autonomous, driverless cars, similar technologies may be considered for urban transportation networks, where cars coordinate their motion to reduce delays while enhancing safety [25, 33]. Consider, for example, vehicles arriving near a traffic intersection where a central control system adjusts their speeds to lead them through the intersection as quickly as possible. In the absence of traffic lights, when the

only limiting factor is avoiding collision with other vehicles, the vehicles' motion can be carefully adjusted, to ensure that the intersection is crossed as quickly as possible. This may require, for instance, that the vehicles traveling in the same lane form series of small platoons, whereas the vehicles from different lanes cross the intersection in close proximity to one another, while traveling at high speeds.

The importance of this problem has not gone unnoticed. In fact, there is a significant and growing body of literature on the problem of motion coordination through all-autonomous intersections, which we will consider in the following section.

## 1.2 Related Work on All-Autonomous Intersections

Coordination control systems based on auction algorithms [8], multi-agent simulation [17, 31], genetic algorithms [24] and token-based approaches [20] were proposed very recently, and implementations using vehicle-to-vehicle communication networks were considered [4]. Similar problems were also studied from the perspective of hybrid control systems [9, 15] and air traffic management [21, 23].

All of the aforementioned approaches use computational experiments to show that the proposed algorithms are safe and they provide good performance. In many cases, the improvement on all-autonomous intersections is so drastic that it leads to several orders of magnitude reduction in average delay for vehicles crossing the intersection. Although the results of the computational experiments are very encouraging, to the best of our knowledge, optimal algorithms and mathematically rigorous performance bounds are not known.

The control of all-autonomous intersections is a more recent interest in the field of intersection management. Nevertheless, there are some important developments despite this relatively new field. Before we explore developments in all-autonomous intersection management, we consider some important work regarding the motion planning of vehicles approaching semi-autonomous intersections, *i.e.*, intersections populated with a combination of human drivers and autonomous vehicles.

Onieva in 2012 developed a controller for a vehicle approaching a two-lane intersection populated by human drivers that is safe, illustrated in Figure 1-3. The human drivers do not

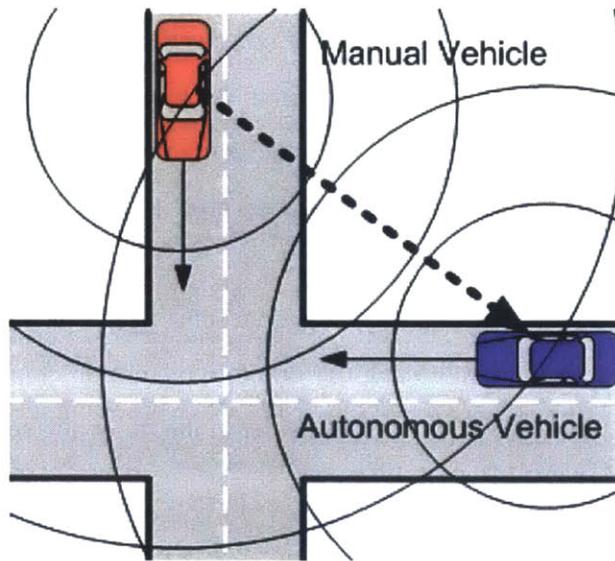


Figure 1-3: Illustration of an autonomous vehicle approaching an intersection with an uncooperative manual driver [24].

cooperate with the vehicle, but neither are they antagonistic; they simply do not take deviate from their nominal trajectory due to the presence of the autonomous vehicle. Although safe, this semi-autonomous intersection leads to suboptimal performance, since one does not consider how to motion plan vehicles in a global sense, and therefore, vehicle cooperation to achieve some global goal, such as minimizing delay, is not explored.

Also, in 2012, Colombo and Vecchio developed a computationally efficient, hybrid controller that preserves the safety of vehicles approaching an intersection. Their framework allows for quite general vehicular models; however, again like Onieva, they focus on demonstrating safety and do not emphasize performance. Moreover, their hybrid controller is designed to augment the actual vehicle controller, *i.e.*, the control action for the vehicle is generated by a nominal controller and then fed through the hybrid controller to ensure safety. Although this methodology allows one to consider general motion planners for vehicles, even quite aggressive ones, there is no cooperation between vehicles either in the same lane or in a different lane, again, possibly restricting the overall performance of the system.

The previous works mentioned focused on developing safe controllers for vehicles approaching a busy intersection. They did not consider how to globally motion plan vehicles. Now, we present current work in intersection management. In these works, the authors

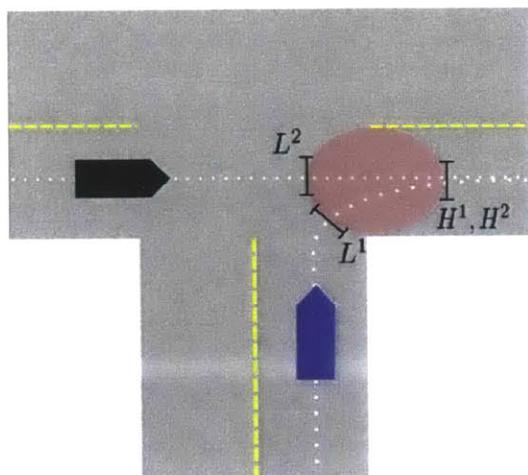


Figure 1-4: Illustration of an autonomous vehicle with a computationally efficient, hybrid controller that preserves safety at intersection [9].

consider the problem of scheduling vehicles as they approach an intersection, although they make use of a simplified vehicular model.

Mao in 2001 considered intersecting deterministic streams of aircraft [23]. He discovered a criterion on the aircrafts' ability to move laterally away from the stream that guarantees safety of the intersecting streams. Note that the incoming streams are deterministic. We are interested in optimally performing algorithms for the case of stochastically arriving of vehicles, which allows us to model more appropriately vehicles in a real environment.

Stone and Dresner in 2009 developed an algorithm to schedule vehicles approaching an autonomous intersection [17]. The intersection region is spatially discretized. Vehicles approaching the intersection region reserve certain slots for certain periods of time, as they request to continue straight, or turn left or right. Not only do the authors demonstrate the safety of their algorithm, they demonstrate the high performance of their algorithm in simulations. However due to the complexity of the algorithm, it is difficult to establish rigorous bounds on the performance of the algorithm; thus, the authors rely mainly on simulations to analyze their algorithm.

The algorithm we propose in this thesis schedules vehicles as they approach the intersection, while also motion planning their trajectories. However, we do consider simpler dynamics. In later work, we wish to extend the algorithm developed in this thesis to more general dynamics.

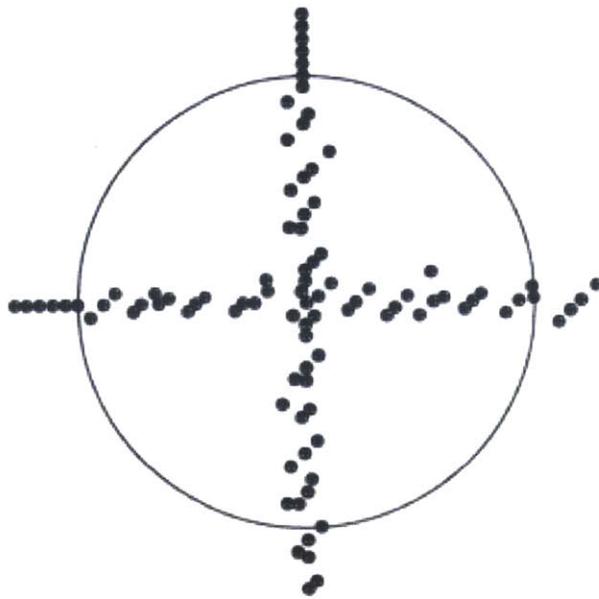


Figure 1-5: Two intersecting streams of aircraft [23].

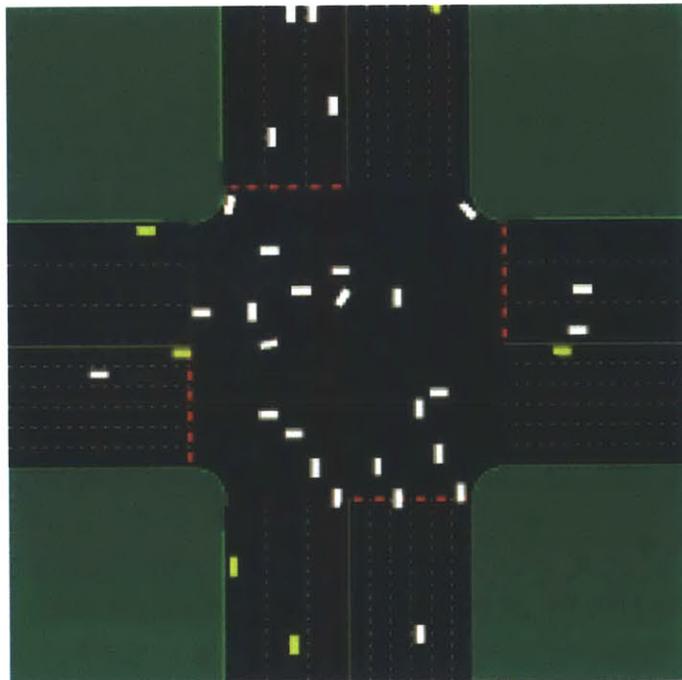


Figure 1-6: AIM algorithm for scheduling vehicles approaching an autonomous intersection [17].

### 1.3 Thesis Contributions

The main contribution of this thesis is a motion coordination algorithm for all-autonomous intersections. Our algorithm is based on the polling systems literature, and it provides provable guarantees on safety and performance. In fact, our algorithm uses a particular polling policy at its core, and its performance is tied to the performance of this polling policy. Roughly speaking, we show that no collisions occur at all times, and moreover, the delay each customer experiences is bounded by the delay in a corresponding polling policy executed on a traditional polling system. The latter result implies an analytical bound on the average delay.

Let us note that the polling systems literature is fairly rich [6, 7, 19, 28, 32]. Motivated by applications in communication systems, transportation systems, and manufacturing, the literature has flourished during the last few decades. Analytical expressions were derived for a range of polling policies [7, 28, 32], and these foundational results have been utilized in a variety of application domains [6, 19]. The traditional applications of polling systems include urban traffic flows [6].

However, to the best of our knowledge, the applications of polling systems in the context of all-autonomous traffic intersections is novel. Furthermore, the problem formulation presented in this thesis can be generalized leading to a new class of polling systems where the customers are subject to differential constraints, which may be interesting on its own right. In this case, the customers must be “steered” to a suitable state before they can be serviced. Our results imply that, in a certain class of such polling systems, the differential constraints can be managed, *i.e.*, the differentially-constrained polling system can achieve the same performance that its counterpart with no differential constraints achieves.

Although this thesis focuses on applications in urban transportation, let us emphasize that potential application areas also include air transportation as well as warehouse automation and manufacturing. In particular, trajectory-based operations considered for the NextGen air transportation system (see, *e.g.*, [22]) by the Federal Aviation Administration in the U.S. will enable trajectory planning and precision execution for aircraft. An effective use of the shared airspace may be possible by setting up virtual roads and intersec-

tions, where the aircraft coordinate their motion for increased performance. Furthermore, autonomous robotic vehicles servicing warehouses, factories, and transportation hubs (*e.g.*, airports, seaports, train stations, *etc.*) may enable efficient transportation of goods and people, with the help of effective motion coordination algorithms.

# Chapter 2

## Problem Definition

We are interested in the problem of motion coordination through a traffic intersection, as depicted in Figure 2-1. Consider a traffic intersection where two orthogonal lanes intersect. Suppose each vehicle is subject to second order dynamics of the following form:

$$\ddot{x}(t) = u(t), \tag{2.1}$$

where  $x(t)$  denotes the position of the front bumper of the vehicle,  $0 \leq \dot{x}(t) \leq v_m$  is the maximum velocity constraint, and  $|u(t)| \leq a_m$  is the maximum acceleration constraint. The region where the two lanes intersect is called the *intersection region*. The portion of the road within a distance of  $L$  to the intersection region is called the *control region*. We assume that, once a vehicle is inside the control region, it is controlled by a central control system. In other words, the input signal  $u(t)$  is directly determined by a central control system for all vehicles that are in the control region.

This central control system does not know *a priori* the precise times that each vehicle will arrive at the control region. However, we assume that certain statistics of their inter-arrival times are available. More precisely, we model the arrival times  $\{t_{j,k} : j \in \mathbb{N}\}$  as a suitable stochastic process, where  $t_{j,k}$  is the time that the  $j$ th vehicle enters the control region (front bumper is exactly a distance of  $L$  away from the end of the intersection) from lane  $k \in \{1, 2\}$ . In other words, the  $j$ th vehicle entering the control region from lane  $k$  is at position  $-L$  at time  $t_{j,k}$ , i.e.,  $x_{j,k}(t_{j,k}) = -L$ , where  $x_{j,k}(t)$  denotes the position of this



Figure 2-1: An illustration of a fully autonomous traffic intersection. Autonomous vehicles arriving near the intersection are fully controlled by a central control system. The control system ensures that the vehicles safely pass through the intersection; furthermore, the central control system provides provable guarantees on performance, *e.g.*, an upper bound on the average time it takes a typical vehicle to go through this intersection region.

vehicle at time  $t$ . From this point on, the position  $x_{j,k}(t)$  of the same vehicle is governed according to the dynamics given in Equation (2.1). If the lane number of the  $j$ th vehicle is clear from context, we simplify the notation and drop the lane subscript, *i.e.*,  $x_j$  denotes the trajectory of the  $j$ th vehicle,  $t_j$  denotes the time the  $j$ th vehicle enters the control region, *et cetera*. We assume that the vehicles enter the control region with maximum speed, *i.e.*,  $\dot{x}(t_{j,k}) = v_m$  for all  $j \in \mathbb{N}$  and all  $k \in \{1, 2\}$ . To simplify notation, the  $j$ th vehicle that enters lane  $k$  is sometimes denoted as vehicle  $(j, k)$ .

We represent each vehicle as a two-dimensional, rectangular rigid body with length  $l$  and width  $w$ . The position of this rigid body is encoded with respect to one of the corners of the intersection region. See Figure 2-2. The position of the rigid body represents the center of the front bumper of the vehicle. The orientation of this rigid body depends on which lane the vehicle is traveling in. To formalize, let us define the rigid body at position  $z \in \mathbb{R}$  in lane  $k$  with  $R(z, k) \subset \mathbb{R}^2$ , *i.e.*,

$$R(z, 1) := \{(y_1, y_2) \in \mathbb{R} : z - l < y_1 < z, 0 < y_2 < w\},$$

and

$$R(z, 2) := \{(y_1, y_2) \in \mathbb{R} : 0 < y_1 < w, z - l < y_2 < z\}.$$

Then, the  $j$ th vehicle entering lane  $k$  at time  $t$  is represented by  $R(x_{j,k}(t), k)$ .

Let  $I_k(t)$  denote the indices for all vehicles that are inside the control region at time  $t$

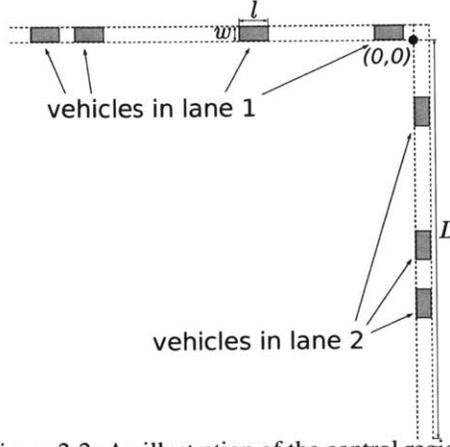


Figure 2-2: An illustration of the control region.

and in lane  $k$ . For instance, if the 3rd, 4th, and 5th vehicles are in lane 1 at time  $t$ , then we have  $I_1(t) = \{3, 4, 5\}$ . Clearly,  $I_k(t)$  is a set of consecutive natural numbers for all  $t \geq 0$  and all  $k \in \{1, 2\}$ .

**Definition 2.0.1 (Safety)** *The control region is said to be safe at time  $t \in \mathbb{R}_{\geq 0}$ , if there are no pairwise collisions among the vehicles, i.e.,  $R(x_{j,k}(t), k) \cap R(x_{i,l}(t), l) = \emptyset$  for all  $j \in I_k$ , all  $i \in I_l$  and all  $k, l \in \{1, 2\}$ .*

For each vehicle, we define the delay incurred in transitioning through the intersection as follows. Recall that  $t_{j,k}$  is the time that the  $j$ th vehicle enters lane  $k$ . Let  $T_{j,k}$  denote the time that the same vehicle completely exits the intersection region, that is, the rear bumper of the vehicle is outside the intersection region. More precisely,  $T_{j,k}$  is the time instance for which  $x_{j,k}(T_{j,k}) = l + w$ . Then, the time it takes for the same vehicle to transition the control region is  $T_{j,k} - t_{j,k}$ . We define the delay as the difference between this time and the time it takes the vehicle to traverse the control region had it been empty. Consider Figure 2-3 for a graphical interpretation of a vehicle's delay.

**Definition 2.0.2 (Delay)** *The delay of vehicle  $(j, k)$  is defined as*

$$D_{j,k} := (T_{j,k} - t_{j,k}) - \frac{L + l + w}{v_m}.$$

$D_{j,k}$  denotes the delay and  $(L + w + l)/v_m$  is the time it would have taken the vehicle to traverse the control region had there been no vehicles in the control region.

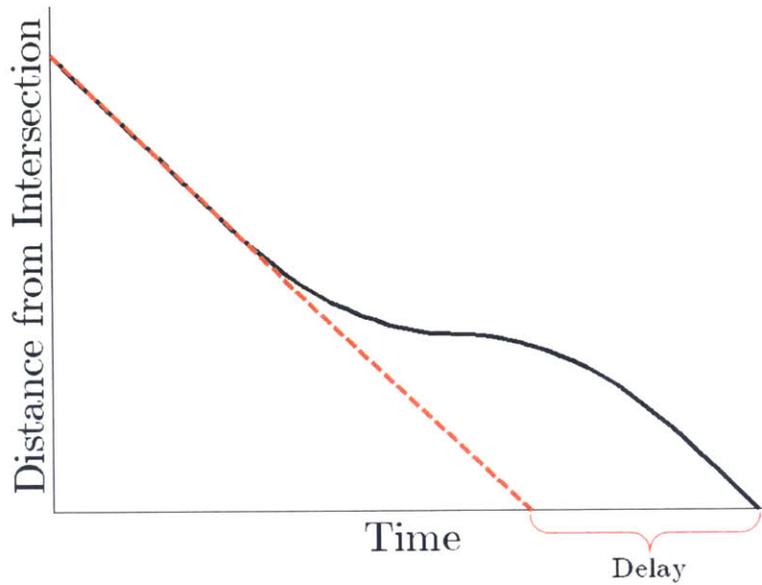


Figure 2-3: Graphical interpretation of the delay of a vehicle as given by Definition 2.0.2.

We are interested in developing coordination algorithms that govern this central controller such that both performance (for example, in terms of delay) and safety (avoiding collisions between vehicles) are simultaneously guaranteed. We will discuss such algorithms in the next section.

At this point, we tacitly leave out an important part of the problem formulation: we do not specify the distribution of the stochastic processes  $\{t_{j,k} : j \in \mathbb{N}\}$ . We will identify this as a part of our model and assumptions in Chapter 4.

# Chapter 3

## Control Policy

In this chapter, we propose a control policy for the intersection coordination problem we introduced in Chapter 2. This policy is based on the polling policies for polling systems [28]. Before describing the policy, we briefly introduce queueing and polling systems and their analysis below, in order to provide a framework for determining how to globally schedule vehicles in an efficient manner. This framework is rich enough that we can apply it to our intersection management problem as defined in Chapter 2, providing us our performance bounds, while also simple enough to simulate in a straightforward manner.

### 3.1 Background on Queueing and Polling Systems

Loosely speaking, queueing theory studies the behavior of wait lines and queues. One of the widely studied mathematical models can be described as follows. Suppose customers arrive at the system, where their requests are processed by one server. Let  $t_j$  denote the arrival time of the  $j$ th customer. Let  $s_j$  denote the time it takes for this customer to be serviced by the server, once it gets the server's attention. Often, both  $\{t_j : j \in \mathbb{N}\}$  and  $\{s_j : j \in \mathbb{N}\}$  are stochastic processes. Suppose the customers are serviced by the server one at a time, in the order that they arrive. Then, given the statistics of arrival times and service times, what is the time average queue length (number of customers who have arrived, but have not yet been serviced) or what is the average wait time for a typical customer? Queueing theory aims to answer these questions for a variety of queueing models. The queueing theory

literature has found profound applications in a number of domains [11], including urban traffic [18].

Polling systems are extensions of queueing systems, where the servers service multiple sets of customers arriving in different queues. The server may choose to serve a customer from any queue. However, the server must pay a set-up cost each time it serves customers coming from a queue that is different than the queue of the previous customer. This situation may arise, for example, in manufacturing machines that need to change their tooling each time they switch to processing a different kind of manufacturing good, where changing the tooling may require time.

Central to polling systems is a controller that decides which queue to serve next. This decision is a determining factor for the performance of the system, for instance, in terms of the average delay or the queue length. In either case, the control policy must trade off the following two. On the one hand, it should switch between different queues often enough, in order to ensure customers in one queue do not wait too long for the server to process customers in a different queue; on the other hand, the server should not switch too often in order not to incur too much set-up cost.

Let us describe a widely-studied polling system example that extends the queueing system example we discussed earlier.

Consider a polling system with two queues and one server. Let  $t_{j,k}$  denote the time that customer  $j$  arrives in queue  $k$ , and let  $s_{j,k}$  denote the amount of time it takes the server to service this customer, where  $k \in \{1, 2\}$ . Both  $\{t_{j,k} : j \in \mathbb{N}\}$  and  $\{s_{j,k} : j \in \mathbb{N}\}$  are stochastic processes, for  $k = 1, 2$ . The customers are serviced by the server one at a time. Switching from one queue to the other requires a set-up time, say  $r$ , which is also a random variable. That is, if the server last serviced a customer from queue 1 and it decides to service a customer from queue 2 next, then a set-up time of  $r$  time units is incurred in addition to the service time of the customers; no set-up time is required, if the server decides to continue servicing customers from queue 1. Strictly speaking, we must identify how exactly the set-up time cost incurs when all queues are empty. For example, in the wait-and-see system (see [28]), if all queues are empty, then the server waits in the queue it last served. If a customer from this queue arrives, then no set-up cost is incurred.

However, if a customer from a different queue arrives, then the server pays the set-up cost and switches to the other queue. After each service completion, the server must make a decision: continue serving customers from the same queue or switch to the next queue.

Polling systems theory analyzes the performance of various polling policies [28]. Some popular examples include the *exhaustive policy*, *gated policy*, and the *k-limited policy*. In the *exhaustive policy*, the server continues to service customers from the same queue until that queue is empty. In the *gated policy*, right after the server switches over to a new queue, it takes a snapshot of this queue; and the server services only those customers in the snapshot, but not the customers that arrive after the snapshot is taken. In the *k-limited policy*, the server services the customers in the same queue until either  $k$  customers are serviced or all customers in the queue are serviced, whichever comes first. Once these customers are serviced, the server switches to the next queue. These policies can be formalized easily. We refer the reader, for example, to [28].

Polling systems literature has been active for several years [6]. The existing literature, including the analysis of these policies for a large class of similar polling models [32], focuses on the case when the interarrival times of the customers have independent identical memoryless distribution, *i.e.*, the arrival times process  $\{t_{j,k} : j \in \mathbb{N}\}$  is a Poisson process for all queues  $k \in \{1, 2\}$ . For example, the necessary and sufficient conditions for stability, the expected delay, and the steady state queue length are known for all of the three policies considered above [28]. In fact, these values can be computed when the number of queues is more than two and the intensity of arrival times is different across queues [28].

Unfortunately, it is analytically challenging to find optimal policies [28]. However, approximation results for limiting cases are available. For example, on the one hand, the exhaustive policy is known to induce lower delay when compared to the gated polling system in a light-load regime, *i.e.*, when the intensities of the arrival times are close to zero for both queues [28]; on the other hand, the gated policy is known to be better than the exhaustive one in the heavy-load regime, *i.e.*, when the load is close to instability [28, 32].

## 3.2 Simulating Polling Systems Behavior

The coordination algorithms we present below heavily rely on the polling systems policies. In particular, in a number of places we simulate the behavior of a polling policy forward in time. We devote this section to formalizing this procedure.

Consider a polling system with two queues and deterministic service and set-up times. That is, the customers arrive at times  $\{t_{j,k} : j \in \mathbb{N}\}$ , where  $k \in \{1, 2\}$ ; their service requires  $s$  time units, and the server requires  $r$  time units to switch queues. The arrival times of the customers are stochastic processes. However, suppose the variables  $s$  are  $r$  the fixed for all the customers. The coordination algorithm we present in Section 3.3 relies on simulating the behavior of a polling system that has fixed service time  $s$  and fixed set-up time  $r$ .

For notational convenience, we represent a polling system with the symbol  $\mathcal{P}$ . The algorithms we describe below interact with a polling system through two procedures. The procedure  $\mathcal{P}.\text{AddToQueue}(k)$  adds one customer to queue  $k$ . The  $\mathcal{P}.\text{Simulate}()$  procedure simulates the behavior of the polling system, assuming no additional customers arrive. More precisely, the  $\mathcal{P}.\text{Simulate}()$  returns two sequences of time instances, namely  $\mathcal{T}_1 = (\tau_{i_1,1}, \tau_{i_2,1}, \dots, \tau_{i_{n_1},1})$  and  $\mathcal{T}_2 = (\tau_{j_1,2}, \tau_{j_2,2}, \dots, \tau_{j_{n_2},2})$ , where  $\tau_{j,k}$  is the time that the server is scheduled to begin servicing the  $j$ th customer in queue  $k$ . Note that the behavior of the polling system  $\mathcal{P}$  depends on its polling policy (e.g., exhaustive, gated,  $k$ -limited, *et cetera*), the service time  $s$ , the set-up time  $r$ , the number of customers in the two queues, and the time the server began serving the current customer (if the server is currently serving any customers) or the time that the set-up operation started (if the server is currently switching over to the next queue). Given the values of all these variables, the  $\mathcal{P}.\text{Simulate}()$  procedure is well-defined in the sense that the sets  $\mathcal{T}_1$  and  $\mathcal{T}_2$  are uniquely determined, if the service time  $s$  and the set-up time  $r$  are fixed (not random).

Let  $m$  and  $n$  denote the number of customers in queues 1 and 2, respectively. Let  $t$  denote the time since the current customer's service has started. Let  $\text{Simulate}(m, n, t)$  denote the procedure that simulates the behavior of the polling assuming no additional customers arrive. More precisely, the  $\text{Simulate}(m, n, t)$  procedure returns the time instances

$\tau_{i_1,1}, \tau_{i_2,1}, \dots, \tau_{i_n,1}$  and  $\tau_{j_1,2}, \tau_{j_2,2}, \dots, \tau_{j_m,2}$ , where  $\tau_{j,k}$  is the time that  $j$ th customer arriving in the  $k$ th queue is scheduled to start being serviced by the server.

Notice that, when the service time  $s$  and the set-up time  $r$  are fixed (not random), then there is a unique set of  $\tau_{j,k}$ 's; hence, the `Simulate`( $m, n, t$ ) procedure is well defined.

### 3.3 The Intersection Coordination Algorithm

We propose an event-triggered coordination algorithm that plans the motions of all vehicles that enter the control region as described in Chapter 2. More precisely, the control algorithm computes  $x_{j,k}$ , for all vehicles that are in the control region, such that the trajectories  $x_{j,k}$  are dynamically feasible and no two vehicles collide. The coordination algorithm is event-triggered in the sense that the trajectories  $x_{j,k}$  are updated each time a new vehicle arrives at the control region.

The core procedure embedded in this event-triggered coordination algorithm is presented in Algorithm 1. Each time a new customer arrives in queue  $k$ , this procedure is triggered. The procedure computes and returns the trajectories  $x_{i,1}$  for all  $i \in \{i_1, i_2, \dots, i_{n_1}\}$  and  $x_{j,2}$  and all  $j \in \{j_1, j_2, \dots, j_{n_2}\}$ , each time a new vehicle arrives at the control region.

Before presenting the coordination algorithm, let us introduce the following motion planning procedure. The `MotionSynthesize` procedure generates a trajectory for each vehicle given the time this vehicle must reach the intersection region and the trajectory of the vehicle in front of it. Furthermore, the trajectory is designed such that the vehicle stays as closely as possible to the intersection region at all times. Denote  $t_0$  as the arrival time of this vehicle, and  $t_f$  as the time to reach the intersection region. Suppose the trajectory of the vehicle in front is denoted by  $x' : [t'_0, t'_f] \rightarrow \mathbb{R}$ , where  $t'_0$  is its arrival time in the control region and  $t'_f$  is the time it is scheduled to reach the intersection region. Then, the `MotionSynthesize` procedure computes a new trajectory such that the two vehicles do not collide and the new trajectory reaches the intersection at time  $t_f$  with speed  $v_m$ . More

precisely, we define:

$$\begin{aligned}
& \text{MotionSynthesize}(x', t_f) := \\
& \arg \min_{x: [t_0, \tau] \rightarrow \mathbb{R}} \int_{t_0}^{t_f} |x(t)| dt \\
& \text{subject to } \ddot{x}(t) = u(t), \text{ for all } t \in [t_0, t_f]; \\
& \quad 0 \leq \dot{x}(t) \leq v_m, \text{ for all } t \in [t_0, t_f]; \\
& \quad |u(t)| \leq a_m, \text{ for all } t \in [t_0, t_f]; \\
& \quad |x(t) - x'(t)| \geq l, \text{ for all } t \in [t_0, t_f]; \\
& \quad x(t_0) = -L; \quad \dot{x}(t_0) = v_m; \\
& \quad x(t_f) = 0; \quad \dot{x}(t_f) = v_m,
\end{aligned}$$

where  $v_m$  and  $a_m$  are the maximum velocity and the maximum acceleration of the vehicles,  $l$  is the length of the vehicle, and  $L$  is the length of the road in the control region.

Then, the coordination algorithm works as follows. The algorithm emulates the behavior of a polling system denoted by  $\mathcal{P}$ . We set this policy to have fixed (non-random) service time  $s = l/v_m$  and set-up time  $r = w/v_m$ . Note that the service time  $s$  is precisely the amount of time from when a vehicle's front bumper enters the intersection region to when its rear bumper leaves the control region assuming the vehicle is traveling at  $v_m$ . The set-up time on the other hand is the amount of time from when the vehicle's rear bumper leaves the control region to when its rear bumper exits the intersection region. (See Figure 3-1.) This polling system may be governed by almost any polling policy that satisfies some mild technical assumptions. (See Assumption 4.2.1.) Each time a new customer arrives in the control region from lane  $k$ , the procedure  $\text{AddToQueue}(k)$  (see Algorithm 1) is triggered. This procedure first adds into queue  $k$  of polling system  $\mathcal{P}$  one customer that represents the newly arriving customer (Line 1). Then, the procedure simulates the polling system forward in time assuming no new customers will arrive (Line 2). The result of the simulation is a sequence of times, namely  $\mathcal{T}_1$  and  $\mathcal{T}_2$ . Let us denote the element  $\tau_{j,k}$  of  $\mathcal{T}_k$  by  $\mathcal{T}_k(j)$ . Finally, the procedure generates a trajectory using the  $\text{MotionSynthesize}$

```

1  $\mathcal{P}.$ AddToQueue( $k$ );
2  $(\mathcal{T}_1, \mathcal{T}_2) \leftarrow \mathcal{P}.$ Simulate();
3 for  $k = 1, 2$  do
4   for  $i = 1, 2, \dots, n_k$  do
5      $\tau_{j,k} \leftarrow \mathcal{T}_k(i)$ ;
6      $x_{j,k} \leftarrow \text{MotionSynthesize}(x_{i-1,k}, \tau_{j,k} + \frac{L}{v_m})$ ;
7   end
8 end

```

**Algorithm 1:**  $\text{NewArrival}(k)$  procedure is triggered when a new vehicle arrives in the control region from lane  $k$ .

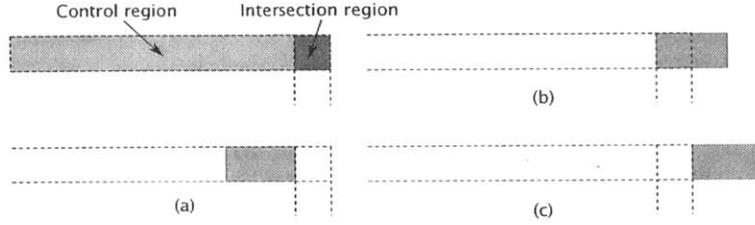


Figure 3-1: We show the three time instances that are described in the text. In (a), the front bumper of the vehicle is entering the intersection region. In (b), the rear bumper is leaving the control region and entering the intersection region. In (c), the rear bumper is leaving the intersection region. In the figure above, the control region (shaded light grey) and the intersection region (shaded in dark grey) are shown.

procedure for each vehicle such that the  $j$ th vehicle in lane  $k$  is scheduled to reach the intersection at time  $\tau_{j,k} + L/v_m$  while avoiding collision with the vehicle in front (Line 6). To be precise, define  $x_{0,k}(t) = l$  for all  $t \in \mathbb{R}_{\geq 0}$  and  $k \in \{1, 2\}$ ; and the  $\text{MotionSynthesize}$  procedure is used to compute  $x_{j,k}$  for all  $j \neq 0$ .

We emphasize that the coordination algorithm depends on the polling policy that governs the polling system  $\mathcal{P}$ . In the next chapter, we show that safety is guaranteed for a wide range of polling policies and that performance can be bounded with respect to the performance of the polling system  $\mathcal{P}$ , which in turn depends on the said polling policy.



# Chapter 4

## Analysis

In this section, we show that the coordination algorithm presented in the previous chapter has two important properties. First, the algorithm is safe in the sense that no collisions occur in the control region. (See Definition 2.0.1.) Second, the coordination algorithm provides good performance. Recall that the coordination algorithm simulates a polling system. We show that the additional delay incurred in traversing the control region is no more than the delay that would incur in servicing a customer in the corresponding polling system, independently of the polling policy that governs the system.

These guarantees hold under certain assumptions. In what follows, we present a class of stochastic process models for vehicle arrival times and two important assumptions. Subsequently, we state and prove our theoretical results, including the key lemma and its corollaries.

### 4.1 Arrival time model

First, we model the arrival times  $\{t_{j,k} : j \in \mathbb{N}\}$  as a hard-core stochastic point process on the non-negative real line [26] such that the  $t_{j+1,k} - t_{j,k} \geq l/v_m$  for all  $j \in \mathbb{N}$  and  $k \in \{1, 2\}$ .<sup>1</sup> The inequality guarantees that no two vehicles are in collision at the time of arrival. A widely studied hard-core stochastic point process, which we also utilize here,

---

<sup>1</sup>Roughly speaking, hard-core point processes are those ensure that the distance between its points are lower bounded by a certain number; in other words, its points are no closer than a certain distance. We refer the reader to [26] for more information.

is the Matérn process [5, 27, 29] generated by thinning a Poisson process. First, a marked Poisson point process is generated on the non-negative real line, where each point  $t_i$  is marked with an independent uniformly random real number between zero and one denoted by  $m(t_i)$ . Subsequently, any point  $t_i$  that satisfies the following is deleted: there exists another  $t_j$  such that the distance between  $t_i$  and  $t_j$  is no more than  $d$  and  $m(t_j) > m(t_i)$ . It is clear that no two points are within a distance of less than  $d$ . Hence, the Matérn process is a hard-core point process.

In addition, we model the effect of over-crowding in the control region to rule out cases that trivially contradict safety. More precisely, we tacitly assume that, if vehicle  $j$  arrives in lane  $k$  at time  $t_{j,k}$  and at this time there is no input that saves it from hitting the car in front, then the same vehicle chooses not to enter the intersection.<sup>2</sup> Let us emphasize that this assumption does not guarantee safety immediately. That there exists some path that does not lead to a collision does not immediately imply that there is a path that is both safe and provides performance guarantees. In other words, this assumption does not render the problem trivial. Without such an assumption, however, the system is (trivially) unsafe. We emphasize that it is impossible to guarantee safety without this assumption. That is, without it, one is faced with trivial cases where any coordination algorithm is unsafe with a non-zero probability. However, clearly, the chance that a vehicle must choose not to enter the intersection is very small, when the road length  $L$  is very large or the intensity of the arrival times is very small (*i.e.*, the light load case). Hence, our analysis applies in these limiting cases, even without this assumption.

## 4.2 Assumptions

Next, we discuss two assumptions. Our first assumption is rather technical. Recall that the behavior of the coordination algorithm depends on the policy of the polling system it simulates. We introduce a regularity assumption on this policy. Roughly speaking, we assume that the policy does *not* respond to the arrival of a customer in queue 1 by favoring

---

<sup>2</sup>For practical purposes, we envision a system with a fork right at the entrance of the control region, and when it is clearly unsafe to enter the control region (in the sense it is impossible to avoid a collision), then the same vehicle takes the exit at the fork and does *not* enter the control region.

the servicing of customers in queue 2, and *vice versa*. The assumption is formalized as follows.

**Assumption 4.2.1 (Regular polling policies)** *Let  $\mathcal{P}$  denote the polling system that the coordination algorithm is using. The polling policy of  $\mathcal{P}$  satisfies the following. At any given time instance, suppose we simulate the polling system to get  $(\mathcal{T}_1, \mathcal{T}_2) \leftarrow \mathcal{P}.\text{Simulate}()$ . Suppose, right at that time instance we first add one customer to queue 1 and simulate the queue afterwards, i.e.,  $\mathcal{P}.\text{AddToQueue}(1)$ ;  $(\mathcal{T}'_1, \mathcal{T}'_2) \leftarrow \mathcal{P}.\text{Simulate}()$  to obtain  $(\mathcal{T}'_1, \mathcal{T}'_2)$ . Then, policy does not schedule customers in queue 2 to an earlier time, i.e.,  $\mathcal{T}'_2(j) \geq \mathcal{T}_2(j)$  for all  $j \in \{j_1, j_2, \dots, j_{n_2}\}$ , and moreover, the policy does not change the scheduled times of the customers in queue 1, i.e.,  $\mathcal{T}'_1(i) = \mathcal{T}_1(i)$  for all  $i \in \{i_1, i_2, \dots, i_{n_1-1}\}$ . The same result also holds for queue 1, whenever a new customer is added to queue 2.*

This assumption is satisfied by many polling policies.

**Proposition 4.2.2** *The exhaustive, gated, and the  $k$ -limited polling policies (see Section 3.3) all satisfy Assumption 4.2.1.*

**Proof** This result can be verified easily for each policy. We verify only for the exhaustive policy. Without loss of generality, suppose a new customer arrives in queue 1. First, suppose the server was processing a customer from queue 1. This would cause the scheduled times for queue 2 customers to be shifted back in time, while it does not affect the scheduled times of customers in queue 1. Second, suppose the server the customer was serving a customer from queue 2 at the time of arrival of the new customer. Then, the servicing times of the customers in queue 1 and 2 do not even change. Hence, Assumption 4.2.1 holds for the exhaustive policy. Similarly, it can be shown that Assumption 4.2.1 is satisfied by gated and  $k$ -limited policies as well, merely by utilizing the same proof technique. ■

We provide our main assumption that the length of the control region is bounded from below by a certain number  $L^*$ . Note that this minimum road length  $L^*$  is a function of the dynamics and geometry of the vehicles.

**Assumption 4.2.3 (The length of the control region)** *The length  $L$  of each of the control regions is bounded from below as follows:  $L \geq \min\{2v_m^2/a_m, v_m^2/a_m + l + w\} \equiv L^*$ .*

As we will show later, this assumption guarantees that the control region is long enough to allow the safe coordination of vehicles while guaranteeing good performance.

### 4.3 Main theoretical results

Our main theoretical results are presented below in Theorems 4.3.2 and 4.3.3. These results are enabled by a key lemma, which we present below.

**Lemma 4.3.1** *Suppose Assumptions 4.2.1 and 4.2.3 hold. Then, each time a new vehicle arrives and Algorithm 1 is called, every call to the MotionSynthesize procedure (Line 6 of Algorithm 1) yields a feasible optimization problem such that the following condition is met. Denote  $x_{j,k}$  as the previous trajectory of the  $j$ th vehicle in lane  $k$ , i.e., the trajectory of vehicle  $(j, k)$  before the latest call to Algorithm 1, and  $\bar{x}_{j,k}$  as the updated trajectory assigned to vehicle  $(j, k)$  via the last call to Algorithm 1. Then, the updated trajectory  $\bar{x}_{j,k}$  agrees with the previous trajectory  $x_{j,k}$  for all time  $t$  before the last call to Algorithm 1, i.e.,  $\bar{x}_{j,k}(t) = x_{j,k}(t)$  for all  $t \leq t_0$ , where  $t_0$  is the arrival time of the newly arrived vehicle.*

Recall that the way we formulated the MotionSynthesize procedure, it solves for the entire trajectory of the vehicle, not just the future portion of its trajectory; therefore, to maintain feasibility of implementation of this algorithm, we must have that the MotionSynthesize procedure does not update the past history of a vehicle's trajectory. One can also formulate the MotionSynthesize procedure so as to solve for just the future part of a vehicle's trajectory, instead of the entire trajectory; however, first note that this is just a reformulation of the procedure as currently posed, and secondly, we find that the current formulation of the MotionSynthesize procedure facilitates some details of the proof of Lemma 4.3.1.

Before providing the proof of this lemma, let us point out two important corollaries, which are our main results.

First, the algorithm guarantees safety surely. In other words, it is guaranteed that no collisions occur as new customers arrive at the control region.

**Theorem 4.3.2 (Safety)** *Suppose Assumptions 4.2.1 and 4.2.3 hold. Then, the control region is safe at all times  $t \geq 0$  in the sense of Definition 2.0.1.*

Second, the delay experienced by the vehicles is bounded by the delay of the corresponding polling system.

**Theorem 4.3.3 (Performance)** *Suppose Assumptions 4.2.1 and 4.2.3 hold. Recall that the delay incurred in transitioning through the control region for vehicle  $j$  in lane  $k$  is denoted by  $D_{j,k}$ . (See Chapter 2.) Let  $W_{j,k}$  denote the wait time of the corresponding customer added to the polling system (see Algorithm 1, Line 1) when vehicle  $j$  arrives in the control region from lane  $k$ . Then, we have the following:*

$$D_{j,k} \leq W_{j,k}, \quad \text{almost surely.}$$

In essence, Theorem 4.3.3 states that the differential constraints that bind the customers as they traverse the control region are irrelevant. The delay is no more than the delay that is incurred in the corresponding polling system. Hence, one can employ any polling policy in the coordination algorithm and guarantee the same performance provided by this policy.

We emphasize that the expected delay is known for many polling policies, in particular the exhaustive and gated policies, when the arrival process is Poisson [28]. Then assuming that the vehicle arrival times is a stochastic Matérn point process, the expected delay time can be bounded as follows.

**Corollary 4.3.4** *Let  $\mathbb{E}[D]$  denote the average delay for a typical vehicle while traversing the control region, when the arrival times of the vehicles is a Matérn process with parameter  $\lambda$ , for some polling policy. Let  $\mathbb{E}[W]$  denote the average delay for a typical customer in a polling system with Poisson arrivals with intensity  $\lambda$ , for the same polling policy. Then,  $\mathbb{E}[D] \leq \mathbb{E}[W]$ .*

The proof of this simple corollary of Theorem 4.3.3 follows directly from the fact that the Matérn process with parameter  $\lambda$  is obtained by thinning the corresponding Poisson process with the same parameter. The value of  $\mathbb{E}[W]$  can be derived for many polling policies [28].

Finally, we provide the proof of the main lemma.

**Lemma 4.3.1** (Sketch) We prove this lemma by invariance. Suppose a vehicle enters the control region at, say, time  $t_0$ . We assume that all the vehicles at time  $t_0$  are following safe and feasible trajectories, which had been generated by Algorithm 1. Now at time  $t_0$ , we step through Algorithm 1, verifying that the vehicles' trajectories are updated to safe and feasible trajectories, as required by Lemma 4.3.1. Without loss of generality, let the incoming vehicle arrive in Lane 2. For ease of presentation, we will denote this newly arrived vehicle from Lane 2 as vehicle  $A$ . Updating the schedule times according to Algorithm 1, we denote  $\tau'_j$  as the updated schedule time of a vehicle  $j$  in the control region and  $\tau_A$  as the newly assigned schedule time of vehicle  $A$ . Similarly, we denote  $\tau_j$  as the previous schedule time of a vehicle  $j$  currently in the control region, *i.e.*, the schedule time that vehicle  $j$  had been assigned before its update at time  $t_0$ . By the regular polling policy condition, the remaining vehicles in Lane 2 keep the same schedule times. Thus, the `MotionSynthesize` procedure engenders the same trajectories as before. Lane 2 vehicles continue on their current trajectories. Next we consider Lane 1 vehicles beginning with the vehicles closest to the intersection region, mirroring the same order determined by Algorithm 1. We show the feasibility of Lane 1 vehicle trajectories that satisfy Lemma 4.3.1 in three steps.

First, we consider vehicles within  $v_m^2/(2a_m)$  of the intersection region. More precisely, suppose vehicle  $j$  is located such that  $|x_j(t_0)| < v_m^2/(2a_m)$ . We claim that the schedule time of vehicle  $j$  does not change as a result of the arrival of vehicle  $A$ , *i.e.*,  $\tau'_j = \tau_j$ . We show this with a proof by contradiction. Suppose that the newly arrived vehicle  $A$  is scheduled before vehicle  $j$ , *i.e.*,  $\tau_A < \tau'_j$ . Note that the earliest time vehicle  $j$ 's updated schedule time can be is after the server services vehicle  $A$ , a duration of  $l/v_m$ , and then switches to Lane 1, a duration of  $w/v_m$ . Thus, we have  $\tau_A + l/v_m + w/v_m \leq \tau'_j$ . But  $\tau'_j = \tau_j + l/v_m$ , since the schedule time of vehicle  $j$  is delayed only by the service time of the newly arrived vehicle  $A$ . Thus, we have established that  $\tau_A < \tau_j$ .

We now show that  $\tau_j + L/v_m \geq \tau_A + L/v_m$ , thus, establishing a contradiction. Note that  $\tau_j + L/v_m$  is the time that vehicle  $j$  will reach the intersection, if it continues to follow the trajectory  $x_j$  it had been assigned before the arrival of vehicle  $A$ . Similarly, we note that  $\tau_A + L/v_m$  is the time that vehicle  $A$  will enter the intersection region, under its trajectory

$x_A$ . Next, we note that vehicle  $j$  is too close to the intersection to come to a full stop and still enter the intersection region at full speed, because  $v_m^2/(2a_m)$  is precisely the minimum distance required for a vehicle to accelerate from a full stop to full speed. The earliest time at which newly arrived vehicle  $A$  can reach the intersection is  $t_0 + L/v_m$ , which may be achieved by traversing the control region at maximum speed. The latest time at which vehicle  $j$  completely exits the intersection region is no more than  $t_0 + v_m/a_m + (l+w)/v_m$ , which corresponds to accelerating from a full stop to full speed, a duration of  $v_m/a_m$ , and then clearing the intersection region at maximum speed, a duration of  $(l+w)/v_m$ . Putting these results together, we have

$$\tau_A + \frac{L}{v_m} \geq t_0 + \frac{L}{v_m} \geq t_0 + \frac{v_m}{a_m} + \frac{l+w}{v_m} \geq \tau_j + \frac{L}{v_m},$$

where the second inequality comes from the minimum road length assumption  $L \geq L^* \geq v_m^2/a_m + l + w$ . Since the schedule times of the vehicles within  $v_m^2/(2a_m)$  have not changed, the `MotionSynthesize` procedure updates these vehicles with their current trajectories, which are safe by assumption.

Thus far, we have shown the feasibility of Algorithm 1 for vehicles that are within a distance of  $v_m^2/(2a_m)$  from the intersection. In the second (and third) steps of the proof this lemma, we consider the vehicles outside this range. More precisely, let vehicle  $j$  be located such that  $|x_j(t_0)| \geq v_m^2/(2a_m)$ . First, we note that there may be a series of vehicles whose schedule times are updated to their previous schedule times. Since pushing back a vehicle's schedule time affects the schedule times of all the vehicles behind it, this series of vehicles must be closer to the intersection than the vehicles whose schedule times were updated to a different time, and, thus, they must all follow one another—except for the first vehicle in this series, which follows a vehicle that is within a distance of  $v_m^2/(2a_m)$  from the intersection. Moreover, the updated trajectories of the vehicles in this series must be the same trajectories these vehicles had been following before the arrival of vehicle  $A$ . Therefore, if vehicle  $j$  has the same schedule time, *i.e.*,  $\tau_j' = \tau_j$ , then the updated trajectory  $x_j'$  of vehicle  $j$  must be the same as the trajectory  $x_j$  it had been previously following, since the `MotionSynthesize` procedure is given the same input trajectory and final time.

Now, suppose vehicle  $j$  has a different updated schedule time, *i.e.*,  $\tau'_j \neq \tau_j$ . We first consider the case in which  $\tau'_{j-1} = \tau_{j-1}$ . This implies that the updated trajectory of vehicle  $j - 1$  is the same as its current trajectory. In other words, we are considering the feasibility of Algorithm 1 for the first vehicle whose updated schedule time differs from its previous schedule time. This implies that the updated trajectory  $x'_j$  of vehicle  $j$  will be different than its previous trajectory  $x_j$ . To show the feasibility of the algorithm, we must construct the updated trajectory  $x'_j$  of vehicle  $j$  and then verify that it is indeed feasible, in the sense that the updated and the previous trajectories agree up until time  $t_0$ , *i.e.*,  $x'_j(t) = x_j(t)$  for any  $t \leq t_0$ . We now state some auxiliary claims that are crucial in the further development of the proof this lemma. Define  $C(x', t_0, t_f)$  as the set of dynamically admissible trajectories that enters the control region at time  $t_0$ , reaches the intersection at time  $t_f$ , and is safe with vehicle  $x'$ . **updated trajectory Claim.** If there exists a trajectory  $\tilde{x}_j$  for vehicle  $j$  that is in the constraint  $C(x', t_0, t_f)$  such that it agrees with the previous trajectory  $x_j$  of vehicle  $j$  up until time  $t_0$ , *i.e.*,  $\tilde{x}_j(t) = x_j(t)$  for any  $t \leq t_0$ , then the trajectory  $x'_j \equiv \arg \min_{x \in C(x', t_0, t_f)} \int_{t_0}^{t_f} |x(t)| dt$  outputted by the `MotionSynthesize` procedure also agrees with  $\tilde{x}_j$  up until time  $t_0$ , *i.e.*,  $x'_j(t) = \tilde{x}_j(t)$  for any  $t \leq t_0$ .

This claim allows us to circumvent directly constructing the updated trajectory  $x'_j$  of vehicle  $j$  and verifying its feasibility, by simply requiring us to show the feasibility of a trajectory  $\tilde{x}_j$ , which in general is easier to construct.

**Claim A.** Let  $C$  be defined as above. Let  $x^*$  be the solution to  $\arg \min_{x \in C} \int_{t_0}^{t_f} |x(t)| dt$ . Then,  $x^*$  has the following property: that for any  $x \in C$  and any  $t \in [t_0, t_f]$ , we have  $|x(t)^*| \leq |x(t)|$ .

**Claim B.** Let  $x_1$  and  $x_2$  be trajectories that have the same initial conditions; however, suppose  $x_1$  reaches the intersection at  $t_1$  and  $x_2$  reaches the intersection at a later time  $t_2$ , *i.e.*,  $t_2 > t_1$ . Then, there exists a safe trajectory from the same initial conditions that reaches the intersection at any time  $t_f \in [t_1, t_2]$ .

**Claim C.** There exists a safe trajectory  $x_l$  such that  $|x_l| \leq \min(|x_1|, |x_2|)$  and the terminal conditions of  $x_l$  are the same as  $x_1$ , *i.e.*,  $x_1$  and  $x_l$  reach the intersection at the same time. Similarly, there exists a safe trajectory  $x_u$  such that  $|x_u| \geq \max(|x_1|, |x_2|)$  and  $x_u$  and  $x_2$  reach the intersection at the same time.

We denote the vehicle directly in front of vehicle  $j$  as vehicle  $j - 1$ .

**Claim 1.** Recall that  $t_0$  is the arrival time of vehicle  $A$ , and  $x_j$  denotes the trajectory that vehicle  $j$  is following at time  $t_0$ , before being assigned an updated trajectory  $x'_j$ . At time  $t_0$ , we step through Algorithm 1, and suppose that the trajectory of vehicle  $j - 1$  is updated to its previous trajectory, *i.e.*,  $x'_{j-1} = x_{j-1}$ , but the updated schedule time of vehicle  $j$  differs from its previous schedule time, *i.e.*,  $\tau'_j \neq \tau_j$ . If vehicle  $j$  can come to a full stop before a distance  $v_m^2/(2a_m)$  from the intersection, *i.e.*,  $\dot{x}(t_0)^2/(2a_m) \leq |x(t_0)| - v_m^2/(2a_m)$ , then vehicle  $j$ 's updated trajectory  $x'_j$  agrees with its previous trajectory  $x_j$  up until time  $t_0$ , *i.e.*,  $x'_j(t) = x_j(t)$  for any  $t \leq t_0$ .

Essentially, Claim 1 states that when Algorithm 1 is triggered, if the state of vehicle  $j$  satisfies a certain condition, then the updated trajectory  $x'_j$  assigned to vehicle  $j$  via `MotionSynthesize` is the same as its previous trajectory  $x_j$  up until the current time.

To establish Claim 1, we first note that  $|x'_j| \geq |x_j|$ , since  $t'_f \geq t_f$ . To show this, suppose not. Then there exists  $t$  such that  $|x'_j(t)| < |x_j(t)|$  and  $\dot{x}'_j(t) > \dot{x}_j(t)$ . By Claim C, there exists a safe trajectory  $x_l$  such that  $x_l \leq \min(|x'_j|, |x_j|)$  and  $x_l$  reaches the intersection at time  $t_f$ . But  $x_l$  is strictly less than  $x'_j$  at time  $t$ , thus,  $x'_j$  cannot be the updated trajectory by Claim A.

Now there exists a safe trajectory  $\bar{x}$  from  $x(t_0)$  and  $\dot{x}(t_0)$  such that vehicle  $j$  crosses the intersection at full speed at time  $t'_f$ , since the state of vehicle  $j$  at  $t_0$  is such that the vehicle can come to a full stop before the intersection and accelerate back to full velocity before entering intersection. By Claim A, the updated trajectory  $x'_j$  must satisfy  $|x'_j| \leq \bar{x}$ . Combining this with  $|x'_j| \geq |x_j|$ , we must have that  $|x'_j(t)| = |x_j(t)|$  for any  $t \leq t_0$ , thus establishing Claim 1.

Now, we show that a vehicle  $j$  whose updated schedule time differs from its previous schedule time, *i.e.*,  $\tau'_j \neq \tau_j$ , always satisfies the necessary condition of Claim 1,  $\dot{x}(t_0)^2/(2a_m) \leq |x(t_0)| - v_m^2/(2a_m)$ . If vehicle  $j$  has position  $|x_j(t_0)| \geq v_m^2/a_m$ , then it automatically satisfies the necessary condition of Claim 1. Next, we consider a vehicle  $j$  with position satisfying  $|x_j(t_0)| \in [v_m^2/(2a_m), v_m^2/a_m)$ , and we establish the necessary condition of Claim 1 by proof by contradiction. Suppose  $\dot{x}(t_0)^2/(2a_m) > |x(t_0)| - v_m^2/(2a_m)$ . Given this assumption, we show that the latest time vehicle  $j$  can possibly reach the inter-

section is strictly less than  $t_0 + L/v_m$ . However, a changed schedule time enables vehicle  $j$  to reach the intersection at least after time  $t_0 + L/v_m$ , and thus we have our contradiction. Let us first establish this latter claim, that vehicle  $j$  reaches the intersection after  $t_0 + L/v_m$ .

First, note that by our invariance assumption, vehicle  $j$  can arrive at the intersection at time  $\tau_j + L/v_m$ . We proceed by showing  $\tau_j > t_0$ . The updated schedule time  $\tau'_j$  of vehicle  $j$  is such that  $\tau'_j = \tau_j + s = \tau_A + s + r$ , where  $\tau_A$  is the schedule time assigned to the newly arrived vehicle  $A$ . But for any newly arrived vehicle, its schedule time  $\tau_A$  cannot be earlier than when it entered the control region, *i.e.*,  $\tau_A \geq t_0$ , from which  $\tau_j > t_0$  follows.

Now we establish that given the state constraint imposed on vehicle  $j$  at time  $t_0$ , *i.e.*,  $\dot{x}(t_0)^2/(2a_m) > |x(t_0)| - v_m^2/(2a_m)$ , vehicle  $j$  cannot possibly reach the intersection after time  $t_0 + L/v_m$ . We present this in the following claim.

**Claim 2.** Let  $v_0$  be the velocity of vehicle  $j$  and  $\xi_0$  be the distance vehicle  $j$  is from the intersection at time  $t_0$ . Suppose that the initial velocity and position satisfy  $v_0^2/(2a_m) > \xi_0 - v_m^2/(2a_m)$ . Then, the latest time vehicle  $j$  can possibly reach the intersection is strictly less than  $t_0 + L/v_m$ .

Denote  $v_l$  as the lowest velocity that vehicle  $j$  can achieve while still reaching the intersection with full speed. Note that the distance needed to decelerate from the current velocity  $v_0$  to a full stop and then accelerate from a full stop to full speed is precisely  $(v_0^2 + v_m^2)/(2a_m)$ . But since  $(v_0^2 + v_m^2)/(2a_m) > \xi_0$ , vehicle  $j$  cannot possibly decelerate to a full stop and still reach the intersection with maximum speed, *i.e.*,  $v_l > 0$ . In order to find the latest time vehicle  $j$  can possibly reach the intersection, we wish to solve the following free terminal time optimal control problem:

$$\begin{aligned} & \max_{x: [t_0, t_f] \rightarrow \mathbb{R}} \quad t_0 + \int_{t_0}^{t_f} dt \\ \text{subject to} \quad & \ddot{x}(t) = u(t), \text{ for all } t \in [t_0, t_f]; \\ & 0 \leq \dot{x}(t) \leq v_m, \text{ for all } t \in [t_0, t_f]; \\ & |u(t)| \leq a_m, \text{ for all } t \in [t_0, t_f]; \\ & x(t_0) = x_0; \quad \dot{x}(t_0) = v_0; \\ & x(t_f) = 0; \quad \dot{x}(t_f) = v_m. \end{aligned}$$

Note that we do not include a safety constraint with vehicle  $j-1$ , *i.e.*,  $|x(t) - x'_{j-1}(t)| \geq l$ , in the optimal control problem formulated above, since it produces a suitable upper bound on the maximum terminal time. Also, due to the difficulty of handling state constraints directly, we further relax this problem to the following optimal control problem, for which a solution is readily available:

$$\begin{aligned} \max_{x:[t_0, t_f] \rightarrow \mathbb{R}} \quad & t_0 + \int_{t_0}^{t_f} dt \\ \text{subject to} \quad & \ddot{x}(t) = u(t), \text{ for all } t \in [t_0, t_f]; \\ & |u(t)| \leq a_m, \text{ for all } t \in [t_0, t_f]; \\ & x(t_0) = x_0; \quad \dot{x}(t_0) = v_0; \\ & x(t_f) = 0; \quad \dot{x}(t_f) = v_m. \end{aligned}$$

The control law for this latter problem is simply bang-bang control, which can be easily verified via Pontryagin's Minimum Principle. Furthermore, the optimal trajectory  $x$  that maximizes the terminal time can be divided into two distinct phases. In the first phase, vehicle  $j$  decelerates to the lowest speed  $v_l$  as quickly as possible, while in the second phase, vehicle  $j$  accelerates to full speed, again, as quickly as possible. This trajectory is summarized in Table 4.1. Note that the velocity of vehicle  $j$  throughout the trajectory satisfies the velocity constraint in the original optimal control problem, *i.e.*,  $0 \leq \dot{x}(t) \leq v_m$  for all  $t \in [t_0, t_f]$ ; therefore, this trajectory must also be the solution to the earlier optimal control problem posed.

Phase	$\Delta t$	$\Delta x$	$\Delta v$	$\ddot{x}$
1	$\frac{v_0 - v_l}{a_m}$	$\frac{v_0^2 - v_l^2}{2a_m}$	$v_l - v_0$	$-a_m$
2	$\frac{v_m - v_l}{a_m}$	$\frac{v_m^2 - v_l^2}{2a_m}$	$v_m - v_l$	$a_m$
<b>Total</b>	$\frac{v_m + v_0 - 2v_l}{a_m}$	$\frac{v_m^2 + v_0^2 - 2v_l^2}{2a_m}$	$v_m - v_0$	

Table 4.1: Latest Schedule Time Trajectory

Now, the latest time at which vehicle  $j$  can possibly reach the intersection is

$$t_0 + \frac{v_m + v_0 - 2v_l}{a_m} < \frac{v_m + v_0}{a_m} \leq 2 \frac{v_m}{a_m} \leq t_0 + \frac{L}{v_m},$$

where the first inequality holds from  $v_l > 0$ , the second inequality from  $v_0 \leq v_m$ , and the last inequality from our assumption on the control region length,  $L \geq 2v_m^2/a_m$ . This concludes the proof of Claim 2.

We have shown that since the updated schedule time  $\tau'_j$  of vehicle  $j$  differs from its previous schedule time  $\tau_j$ , we must have that vehicle  $j$  was following a trajectory  $x_j$  at time  $t_0$  that would allow for it to reach the intersection at a time strictly greater than  $t_0 + L/v_m$ . But due to the state constraint imposed on vehicle  $j$  at time  $t_0$ , we have shown that the latest time that vehicle  $j$  can possibly reach the intersection is strictly less than  $t_0 + L/v_m$ . Thus, the state constraint imposed on vehicle  $j$  cannot hold, establishing that vehicle  $j$  always meets the necessary condition of Claim 1, *i.e.*,  $\dot{x}(t_0)^2/(2a_m) \leq |x(t_0)| - v_m^2/(2a_m)$ .

Before we continue with the final part of the proof of this lemma, we introduce some notation for ease of presentation. Let function  $n_j$  be from the set of vehicles in Lane 1 whose updated schedule times  $\tau'_j$  differ from their previous schedule times  $\tau_j$ , *i.e.*,  $n_j: \{j: \tau'_{j,1} \neq \tau_{j,1}\} \rightarrow \mathbb{N}$ , as follows. Let  $n_j$  be defined as  $n_j = j + 1 - \min_j \{j: \tau'_{j,1} \neq \tau_{j,1}\}$ . We say that vehicle  $j$  is the  $n_j^{\text{th}}$  uncommitted vehicle.

We denote  $F_j$  as the set of all allowable distance and velocity pairs  $(\xi_0, v_0)$  so that the  $n_j^{\text{th}}$  uncommitted vehicle, with velocity  $v_0$  and distance  $\xi_0$  from the intersection at time  $t_0$ , can decelerate to a full stop and then accelerate to full speed, without coming within a distance of  $(n_j - 1)l$  from the intersection, *i.e.*,  $F_j = \{(\xi_0, v_0) \in [0, L] \times [0, v_m] : v_0^2/(2a_m) \leq \xi_0 - v_m^2/(2a_m) - (n_j - 1)l\}$ . Furthermore, we denote  $\partial F_j$  as the left-hand side boundary of the set  $F_j$ , *i.e.*,  $\partial F_j = \{(\xi_0, v_0) \in [0, L] \times [0, v_m] : v_0^2/(2a_m) = \xi_0 - v_m^2/(2a_m) - (n_j - 1)l\}$ .

**Claim 3.** Recall that  $t_0$  is the arrival time of vehicle  $A$ . Denote  $x_j$  the un-updated trajectory of an uncommitted vehicle  $j$ . Then, the state of vehicle  $j$  at time  $t_0$  must satisfy  $(|x(t_0)|, \dot{x}(t_0)) \in F_j$ .

We have already established this claim for the case of  $n_j = 1$ ; this is precisely the necessary condition of Claim 1. We continue proving Claim 3 for a vehicle  $j$  satisfying  $n_j > 1$ . Under its previous trajectory  $x_j$ , vehicle  $j$  arrives at the intersection region at time  $\tau_j + L/v_m$ . Also, directly from our invariance assumption, we have that under its previous

trajectory  $x_j$ , vehicle  $j$  must attain full velocity a distance  $(n_j - 1)l$  from the intersection, in order to reach the intersection at time  $\tau_j + L/v_m$  while also being safe with trajectory  $x_{j-1}$ . Moreover, vehicle  $j$  reaches a distance  $(n_j - 1)l$  from the intersection region at time  $\tau_j + L/v_m - (n_j - 1)s$ , since vehicle  $j$  must traverse the distance  $(n_j - 1)l$  at full speed, in order to ensure safety with trajectory  $x_{j-1}$  of vehicle  $j - 1$ .

Recall that  $\tau_A$  denotes the schedule time of newly arrived vehicle  $A$  and also that the updated schedule time  $\tau'_j$  of vehicle  $j$  satisfies  $\tau_j = \tau'_j - s$ . By starting with the arrival time  $\tau_A$  of vehicle  $A$ , counting its service time  $s$ , the time  $r$  to switch to the next queue, and lastly the servicing of the  $n_j - 1$  vehicles in front of vehicle  $j$ , the updated schedule time  $\tau'_j$  of vehicle  $j$  can be expressed as  $\tau'_j = \tau_A + s + r + (n_j - 1)s$ . Therefore, the time vehicle  $j$  reaches a distance  $(n_j - 1)l$  from the intersection region can also be expressed as  $\tau_A + L/v_m + r$ . But for newly arrived vehicle  $A$ , we have that  $\tau_A \geq t_0$ , from which we have that the time vehicle  $j$  reaches a distance  $(n_j - 1)l$  from the intersection region is strictly later than  $t_0 + L/v_m$ .

We establish Claim 3 by a proof by contradiction. Suppose vehicle  $j$ 's state at time  $t_0$  satisfies  $(|x(t_0)|, \dot{x}(t_0)) \notin F_j$ . By mirroring Claim 2, it can be shown that vehicle  $j$  cannot possibly reach a distance  $(n_j - 1)l$  from the intersection region at full speed later than time  $t_0 + L/v_m$ . Simply note that the lowest velocity  $v_l$  that vehicle  $j$  can achieve is strictly greater than 0, *i.e.*,  $v_l > 0$ . Thus, the latest time that vehicle  $j$  can possibly reach a distance  $(n_j - 1)l$  from the intersection region is  $t_0 + (v_m + v_0 - 2v_l)/(a_m)$ , which is strictly less than  $t_0 + L/v_m$  by our assumption on the length of the control region.

Now, we are ready to present some more notation that will be useful in the remainder of the proof of this lemma. When vehicle  $j$  arrives at the intersection region, its state satisfies  $(0, v_m) \notin F_j$ . But at time  $t_0$ , vehicle  $j$ 's state satisfies  $(|x(t_0)|, \dot{x}(t_0)) \in F_j$ . Thus, there exists a time  $t$  such that vehicle  $j$  satisfies  $(|x_j(t)|, \dot{x}_j(t)) \in \partial F_j$ . We denote  $t_c(x_j)$  the earliest such time, *i.e.*,  $t_c(x_j) = \inf\{t: (|x_j(t)|, \dot{x}_j(t)) \in \partial F_j\}$ . Note that by continuity of the mapping  $t \rightarrow (|x(t)|, \dot{x}(t))$ , we have  $t_c(x_j) \in \partial F_j$ , since  $\partial F_j$  is a closed set. Also, it is more convenient to work with the time that a vehicle arrives at the intersection region rather than with its schedule time. Thus, in the remainder of the proof, we denote  $t_f$  as the time vehicle  $j$  would have reached the intersection region if it had continued following its

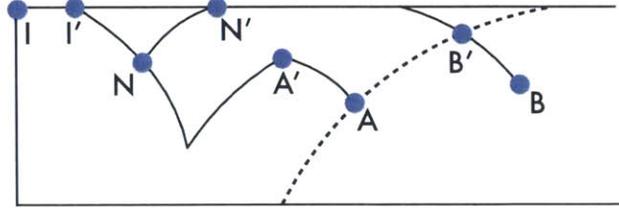


Figure 4-1: Phase portrait for Case 1 where there is no intersection point C. The  $x$ -axis represents distance  $\xi$  from intersection, point  $I$ . The  $y$ -axis plots velocity. The dashed curve is  $\partial F_j$ .

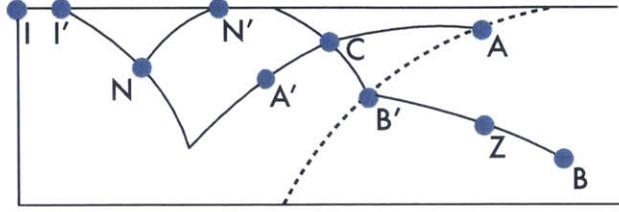


Figure 4-2: Phase portrait for Case 1 where there is an intersection point C. The  $x$ -axis represents distance  $\xi$  from intersection, point  $I$ . The  $y$ -axis plots velocity. The dashed curve is  $\partial F_j$ .

previous trajectory  $x_j$ , *i.e.*,  $t_f = \tau_j + L/v_m$ . Similarly, we denote  $t'_f$  as the time that vehicle  $j$  is now scheduled to reach the intersection region according to its updated schedule time  $\tau'_j$ , *i.e.*,  $t'_f = \tau'_j + L/v_m = \tau_j + s + L/v_m$ .

Claim 1'. If uncommitted vehicle  $j$  has state at time  $t_0$  such that  $(|x_j(t_0)|, \dot{x}_j(t_0)) \in F_j$ , then there exists a trajectory  $\tilde{x}_j$  for vehicle  $j$  such that  $\tilde{x}_j(t) = x_j(t)$  for all time  $t \leq t_0$  and that  $\tilde{x}_j$  reaches the intersection region with full speed at time  $t'_f$ , *i.e.*,  $\tilde{x}_j(t'_f) = 0$  and  $\dot{\tilde{x}}_j(t'_f) = v_m$ .

Claim. Given a feasible and safe trajectory  $\tilde{x}_j$  for vehicle  $j$ , the optimal trajectory  $x'_j$  for vehicle  $j$  satisfies  $x'_j(t) = \tilde{x}_j(t)$  for all  $t \leq t_0$ .

We wish to show that there exists a trajectory  $\tilde{x}_j$  for vehicle  $j$  that agrees with its previous trajectory  $x_j$  up until time  $t_c(x_j)$ , that is safe with the updated trajectory  $x'_{j-1}$ , *i.e.*,  $|x'_j| \geq |x'_{j-1}| + l$ , and that reaches the intersection region with full speed at time  $t'_f$ . We proceed by induction on  $n_j$ . The case  $n_j = 1$  was established earlier in Claim 1. By induction, we assume that  $\tilde{x}_{j-1}(t) = x_{j-1}(t)$  for all time  $t \leq t_c(x_{j-1})$ . By Claim, we have that the updated trajectory  $x'_{j-1}$  for vehicle  $j - 1$  also satisfies  $x'_{j-1}(t) = \tilde{x}_{j-1}(t)$  for all time  $t \leq t_c(x_{j-1})$ . We consider the following 2 cases.

Case 1. Suppose vehicle  $j - 1$  reaches  $\partial F_{j-1}$  first, *i.e.*,  $t_c(x'_{j-1}) \leq t_c(x_j)$ . Let  $t_A$  be the time that vehicle  $j - 1$  arrives at the boundary  $\partial F_{j-1}$ , *i.e.*,  $t_A = t_c(x_{j-1})$ . Let  $t_{B'}$  be the time

that vehicle  $j$  arrives at the boundary  $\partial F_j$ , *i.e.*,  $t_{B'} = t_c(x_j)$ . We construct  $\tilde{x}_j$  as follows. Let  $\tilde{x}_j$  agree with  $x_j$  from time  $t_0$  to time  $t_{B'}$ , *i.e.*,  $\tilde{x}_j(t) = x_j(t)$  for all time  $t \in [t_0, t_{B'}]$ . Up until time  $t_{B'}$ ,  $\tilde{x}_j$  must be safe with  $x'_{j-1}$ . Clearly,  $\tilde{x}_j$  is safe with  $x'_{j-1}$  from time  $t_0$  to time  $t_A$ , since  $|x_j(t)| \geq |x_{j-1}(t)| + l$  for all time  $t \in [t_0, t_A]$  by our invariance assumption. To show safety from time  $t_A$  to time  $t_B$ , consider Figure .

Directly constructing the remainder of trajectory  $\tilde{x}_j$  is difficult. Thus, we take the following approach. We find two trajectories for vehicle  $j$ , which we denote  $\bar{x}$  and  $x_q$ . The trajectory  $\bar{x}$  for vehicle  $j$  extends  $\tilde{x}_j$  in such a way that it is safe with trajectory  $x'_{j-1}$ , *i.e.*,  $|\bar{x}_j(t)| \geq |x'_{j-1}(t)| + l$  for all time  $t$ , and that vehicle  $j$  arrives at the intersection region with full speed after time  $t'_f$ , *i.e.*, there exists time  $t''_f \geq t'_f$ , such that  $\bar{x}(t''_f) = 0$  and  $\dot{\bar{x}}(t''_f) = v_m$ . The trajectory  $x_q$  for vehicle  $j$  is constructed such that it reaches the intersection region with full speed before time  $t'_f$ , even if it may possibly be unsafe with trajectory  $x'_{j-1}$ , *i.e.*, there exists time  $t'''_f < t'_f$  such that  $x_q(t'''_f) = 0$  and  $\dot{x}_q(t'''_f) = v_m$ . Using trajectories  $\bar{x}$  and  $x_q$ , we then show that  $\tilde{x}_j$  exists by a continuity argument.

We construct trajectory  $x_q$  for vehicle  $j$  by extending trajectory  $\tilde{x}_j$  as follows. Vehicle  $j$  accelerates as quickly as possible from time  $t_{B'}$  until it reaches full speed, after which, vehicle  $j$  continues at full speed until it reaches the intersection region. Note that  $|x_q(t)| \leq |x_j(t)|$  for all time  $t$ . But vehicle  $j$  under trajectory  $x_j$  reaches the intersection region at time  $t_f < t'_f$ , from which vehicle  $j$  under trajectory  $x_q$  must reach the intersection region strictly before time  $t'_f$ .

Before we construct trajectory  $\bar{x}$ , we introduce notation that will be used throughout the remainder of the proof. We denote the path of vehicle  $j$  as the curve that vehicle  $j$  traces out on the phase portrait, *i.e.*, a plot of velocity  $v$  versus distance from intersection  $\xi$ . Similarly, we denote the path of vehicle  $j - 1$  as the curve that it traces out on the phase portrait but with a horizontal shift to the right of distance  $l$ . In a sense, we are tracing the rear bumper of vehicle  $j - 1$ , since this allows for a more natural interpretation of safety, when compared to the path of vehicle  $j$ . Thus, vehicle  $j - 1$  “arrives” at the intersection at point  $I'$ . If point  $Q$  is on the phase portrait, we will refer to its coordinates as  $(\xi_Q, v_Q)$ . We denote point  $I$  as the intersection region, and the dashed curve is  $\partial F_j$ . Refer to Figure 4-1 for clarity. As a side note, to uniquely represent a vehicle’s trajectory on a phase portrait,

we also need to keep track of how long the vehicle spends at points of zero velocity in its trajectory.

Now, we are ready to construct trajectory  $\bar{x}$  for vehicle  $j$ . Suppose the path  $P_{BB'I}$  of vehicle  $j$  traced out by trajectory  $x_q$  does not intersect the path  $P_{AA'I'}$  of vehicle  $j - 1$  following  $x'_{j-1}$ , as depicted in Figure 4-1, at any other point than  $I'$ . Then, we define the trajectory generated by path  $P_{BB'AA'I'I}$  as the concatenation of four subtrajectories: the trajectory  $\tilde{x}_j$  from point  $B$  to point  $B'$ , the trajectory traced out by following  $\partial F_j$  from point  $B'$  to point  $A$ , the trajectory  $x'_{j-1}$  represented by path  $P_{AA'I'}$  from point  $A$  to point  $I'$ , and lastly the full speed trajectory from point  $I'$  to point  $I$ . Note that the time for vehicle  $j$  to reach the intersection region under path  $P_{BB'AA'I'I}$  takes at least as long as the time to traverse path  $P_{AA'I'}$ , a duration of  $t_f - t_0$ , plus the time it takes to traverse path  $P_{I'I}$ , a duration of  $s$ ; therefore, under path  $P_{BB'AA'I'I}$ , vehicle  $j$  arrives at the intersection region no earlier than time  $t'_f = t_f + s$ . Moreover, it is not too difficult to see that the trajectory of vehicle  $j$  generated by path  $P_{BB'AA'I'I}$  is safe with trajectory  $x'_{j-1}$ , represented by path  $P_{AA'I'}$  of vehicle  $j - 1$ . Thus, the trajectory generated by path  $P_{BB'AA'I'I}$  satisfies the conditions of  $\bar{x}$ .

Now, we construct trajectory  $\bar{x}$  for vehicle  $j$  when the path of vehicle  $j$  following  $x_q$  does intersect the path  $P_{AA'I'}$  of vehicle  $j - 1$  following  $x'_{j-1}$ , at some point other than  $I'$ . Denote any such point as point  $C$ , as illustrated on the phase portrait in Figure 4-2. We define the trajectory for vehicle  $j$  generated by path  $P_{BB'CA'I'I}$  as the concatenation of three subtrajectories: the trajectory  $\tilde{x}_j$  from point  $B$  to point  $C$ , the trajectory generated by path  $P_{CA'I'}$  from point  $C$  to point  $I'$ , and lastly the full speed trajectory generated by path  $P_{I'I}$ , from point  $I'$  to point  $I$ . Note that path  $P_{BB'CA'I'I}$  satisfies the conditions of  $\bar{x}_j$ . Denote point  $Z$  as the intersection of the vertical line through point  $A$  and the path  $P_{BB'}$ . Note that the path  $P_{AC}$  always lies above  $P_{ZB'C}$ , since point  $C$  is the earliest intersection point of paths  $P_{BB'I}$  and  $P_{AA'I'}$ . Thus, if vehicle  $j$  were to start at point  $Z$  and vehicle  $j - 1$  were to start at  $A$  and follow their respective paths, vehicle  $j$  would always be safe with vehicle  $j - 1$ , i.e.,  $\xi_{AC}(t) \geq \xi_{ZB'C}(t)$ , where  $\xi_{PATH}(t)$  denotes the  $\xi$  coordinate of path  $P_{PATH}$  at time  $t$ . Moreover, since vehicle  $j$  in actuality begins its trajectory at point  $B$  when vehicle  $j - 1$  begins its trajectory at point  $A$ , vehicle  $j$  must first traverse path  $P_{BZ}$

before arriving at point  $Z$ ; thus, vehicle  $j$  under path  $P_{BB'C}$  is safe with vehicle  $j - 1$  under path  $P_{AC}$ . Furthermore, the trajectory of vehicle  $j$  as it traverses path  $P_{BB'CA'I'I}$  from point  $C$  to point  $I$  is safe with the trajectory of vehicle  $j - 1$ , since vehicle  $j$  arrives at point  $C$  later than vehicle  $j - 1$  arrives at point  $C$  and then both follow the same path. Thus, we have shown the safety of  $\bar{x}$  with  $x'_{j-1}$ . Lastly, note that vehicle  $j$  arrives at the intersection region no earlier than time  $t'_f$ . To see this, notice that it takes longer to traverse path  $P_{BB'CA'I'I}$  than path  $P_{ACA'I'I}$ , which requires a duration of  $t_f - t_0$ . To traverse path  $P_{I'I}$  requires a duration of  $s$ , from which we have that vehicle  $j$  arrives at the intersection region no earlier than time  $t'_f = t_f + s$ .

Thus far, we have constructed trajectories  $x_q$  and  $\bar{x}$ . We now show the existence of  $\tilde{x}_j$ . The first step is to show that there exists points  $N$  and  $N'$  such that path  $P_{BB'N'NI'I}$  and path  $P_{AA'I'I}$  are traversed in the same amount of time and are safe with each other.

We first consider the case when there is no intersection point  $C$ , as noted earlier. Suppose we find points  $N$  and  $N'$  such that these two paths are traversed in the same amount of time. Furthermore, suppose that  $N$  and  $N'$  are chosen so that  $P_{BB'N'NI'I}(\xi) \geq P_{AA'I'I}(\xi)$ , for all  $\xi$ . Then, the trajectories generated by these two paths must be safe. Starting at the intersection, note that the upper path is always traveling at a faster velocity than the lower path for each position  $\xi$ , from which we have safety. Thus, it remains to show that points  $N$  and  $N'$  exist such that the two paths are traversed in the same amount of time and that  $P_{BB'N'NI'I}(\xi) \geq P_{AA'I'I}(\xi)$ , for all  $\xi$ .

At each point  $N$  on the path  $P_{AA'I'I}$ , we can accelerate at maximum acceleration until we intersect path  $P_{BB'I'I}$  at some point, which we denote  $N'$ . By construction, path  $P_{BB'N'NI'I}$  is above the path  $P_{AA'I'I}$ , and below path  $P_{BB'I'I}$ . Therefore, the time required for vehicle  $j$  to traverse path  $P_{BB'N'NI'I}$  is lower and upper bounded by the time it takes vehicle  $j$  to traverse  $x_q$  and  $\bar{x}$ , respectively. For any path  $P_{PATH}$ , we define  $\Delta t(P_{PATH})$  as the time duration of traversing path  $P_{PATH}$ . Thus, Note that for each  $N$ , there always exists  $N'$ . Next, the map  $\xi_N \rightarrow \Delta t(P_{BB'N'NI})$  is continuous. Thus, there must exist  $N$  such that  $\Delta t(P_{BB'N'NI}) = \Delta t(P_{AA'I'I}) = t'_f - t_0$ , since the range of the map  $\xi_N \rightarrow \Delta t(P_{BB'N'NI})$  is  $[\Delta t(P_{BB'I}), \Delta t(P_{BB'AA'I'I})] \ni t'_f - t_0$ .

Now, we look at the case where some intersection point  $C$  exists. Verifying safety

in this case is slightly more involved. In this case, we cannot find a path that is always greater than the path traced out by  $x'_{j-1}$ ; however, we can still find a path that is indeed safe with  $x'_{j-1}$ . Suppose we find points  $N$  and  $N'$  such that  $P_{CN'I}(\xi) \leq P_{CN'NI}(\xi)$  and the durations of path  $P_{BB'CN'NI}$  and path  $P_{ACA'NI}$  are equal. Then, safety follows. To see this, consider starting at the intersection and follow the trajectories generated by these paths as time decreases. Since the upper path is always going at a faster velocity, then for any time  $t$  such that vehicle  $j$  is closer to the intersection than point  $C$ , we have  $\xi_{BB'CN'NI}(t) \geq \xi_{ACA'NI}(t)$ . Similarly, starting from time  $t_0$ , since vehicle  $j$  is always traveling at a lower velocity, we have that for any time  $t$  until vehicle  $j$  reaches point  $C$ , we have  $\xi_{BB'CN'NI}(t) \geq \xi_{ACA'NI}(t)$ . Thus, safety of paths  $P_{BB'CN'NI}$  and  $P_{ACA'NI}$  are ensured. Lastly, the map  $\xi_N \rightarrow \Delta t(P_{BB'NI})$  is again continuous; however, we restrict ourselves to only points  $N$  along path  $P_{BB'CI}$  that are closer to the intersection than point  $C$ .

Case 2. Suppose vehicle  $j$  reaches  $\partial F_j$  first, i.e.,  $t_c(x'_{j-1}) \geq t_c(x_j)$ . Then, we show that we can reduce this to Case 1 for some later time  $t'_0 \geq t_0$ . Some time elapses until vehicle  $j-1$  reaches  $\partial F_{j-1}$ , which we denoted time  $t_A$ . Over this time interval  $[t_0, t_A]$ , we define trajectory  $\tilde{x}_j$  as decelerating at full deceleration. First, note that  $|\tilde{x}_j| \geq |x_j|$ . Thus, vehicle  $j-1$  following trajectory  $x'_{j-1}$  until time  $t_A$  is safe with trajectory  $\tilde{x}_j$ . It remains to show that there exists a feasible trajectory for vehicle  $j$  from point  $B'$  so that it reaches the intersection before time  $t'_f$ . With this, note that time  $t_A$  in this case acts like the arrival time  $t_0$  of Case 1. Proof follows by Case 1 argument.

Note that vehicle  $j$  can reach the intersection before time  $t'_f$ , if it were at point  $B$ , at time  $t_A$  (by simply following original trajectory  $x_j$  it arrives at time  $t_f < t'_f$ ). Also, vehicle  $j$  can reach the intersection before time  $t'_f$ , if it were at point  $A'$  at time  $t_A$  (by simply copying the control strategy of vehicle  $j-1$  under trajectory  $x'_{j-1}$ , vehicle  $j$  arrives at the intersection at time  $t'_f$ ). Now, we know that at time  $t_A$ , vehicle  $j$  is on the path  $P_{BA'}$ . Now consider the path from point  $B'$  that accelerates at full acceleration until reaching maximum velocity at point  $D$ , then continuing to the intersection with full speed. Call this path  $P_{B'DI}$ . We lower and upper bound this path as follows. Denote path  $P_{BEDI}$  as the path beginning at point  $B$  and accelerating at maximum acceleration until reaching maximum velocity at point  $E$ ,

then continuing on to the intersection with full speed. We see that  $P_{AI} \leq P_{B'DI} \leq P_{BEDI}$ . Thus, the time duration of path  $P_{B'DI}$  must also be bounded above and below by  $\Delta t(P_{AI})$  and  $\Delta t(P_{BEDI})$ , respectively. But  $\Delta t(P_{BEDI})$  is bounded above by the time vehicle  $j$  traverses to the intersection under trajectory  $x_j$ ; therefore, following path  $P_{BEDI}$ , vehicle  $j$  arrives at the intersection before time  $t_f < t'_f$ . ■



# Chapter 5

## Computational Experiments

In this chapter, we evaluate the proposed coordination algorithm in computational simulations. First, we describe our approach to obtaining numerical approximations to the vehicle trajectories as computed by the `MotionSynthesize` procedure. We then illustrate these trajectories in light-load, medium-load, and heavy-load traffic. Next, we justify our process of thinning out vehicles. Lastly, we compare the performance of our algorithm with an intersection managed by a traditional traffic light with human drivers. We used Matlab to generate the simulation results presented in this chapter, solving the linear programs with Gurobi [14]. Incoming vehicle arrival times were generated by a Matérn process [27]. The vehicle length, width, maximum velocity, and maximum acceleration were taken to be 2 meters, 1 meter, 10 meters/sec, and 4 meters/sec<sup>2</sup>, respectively.

### 5.1 Approximate Vehicle Trajectories

A polling system is initialized with nonzero deterministic service time and switching time as mentioned in Chapter 3. The “wait-and-see” switching policy was chosen to govern the polling system switching dynamics. Approximate trajectories of the vehicles were obtained by discretizing the mathematical program in the `MotionSynthesize` procedure.

This discretization leads to the following linear program:

$$\begin{aligned}
& \max \sum_{i=0}^N x_i \\
& x_{i+1} = x_i + (v_i + v_{i+1}) \cdot \Delta t / 2, \text{ for all } i; \\
& v_{i+1} = v_i + u_i \cdot \Delta t, \text{ for all } i; \\
& 0 \leq v_i \leq v_m, \text{ for all } i; \\
& -a_m \leq u_i \leq a_m, \text{ for all } i; \\
& x_i \leq x'(t_0 + i \cdot \Delta t) - l, \text{ for all } i; \\
& x_0 = -L; \\
& x_N = 0; \\
& v_0 = v_m; \\
& v_n = v_m,
\end{aligned}$$

where  $\Delta t$  is the time step of the time history,  $t_0$  is the arrival time of some vehicle that triggers the algorithm,  $[x_0, \dots, x_N, v_0, \dots, v_N, u_0, \dots, u_{N-1}]^T$  is the state vector, representing the discretized position, velocity and acceleration of the vehicle at time  $t_0 + i \cdot \Delta t$ , and  $x'$  is the trajectory of the vehicle directly in front. Note that the trajectory  $x'$  of the vehicle ahead is interpolated at time  $t_0 + i \cdot \Delta t$ . Expected delays were estimated by running the simulation for a long time and averaging the delay incurred by each vehicle in the system. We find that  $N = 800$  yields sufficiently smooth trajectories. However, in general, the required degree of discretization varies with the control region length. Also, since the vehicle directly in front is discretized, we found that a quadratic interpolation scheme, using the positions as well as the velocities, significantly improves the quality of approximation than using the standard linear interpolation.

In Figure 5-1, we show trajectories of vehicles under an exhaustive policy with increasing load. It can be seen that the platooning behavior emerges when the load becomes substantial. In Figure 5-2, we show trajectories of vehicles under a  $k$ -limited polling policy. In none of these simulations, the MotionSynthesize procedure reported infeasible, which

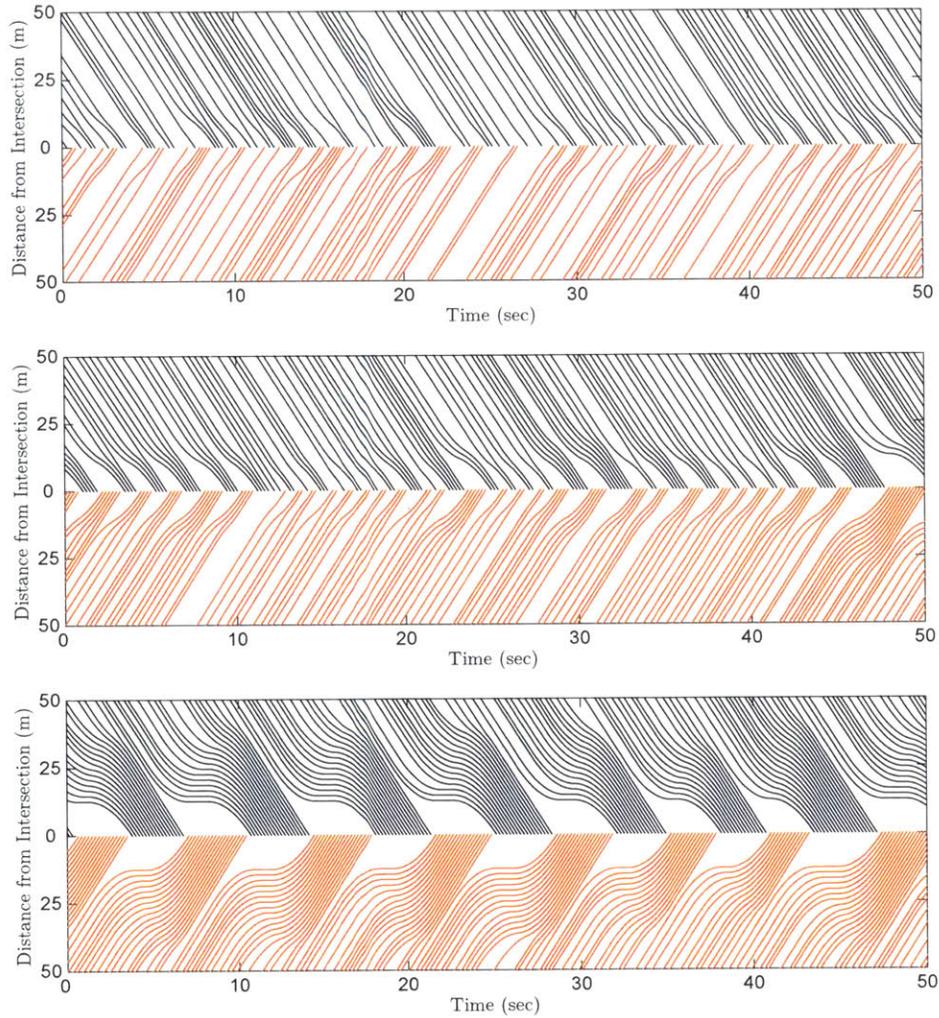


Figure 5-1: The trajectories of the vehicles are shown. The top, middle, and bottom figures are light, medium, and heavy load cases, respectively. The top half of each figure shows the trajectories of vehicles in Lane 1, and the bottom half is the trajectories of vehicles from Lane 2. Each figure shows a small window of time after the system reaches steady state.

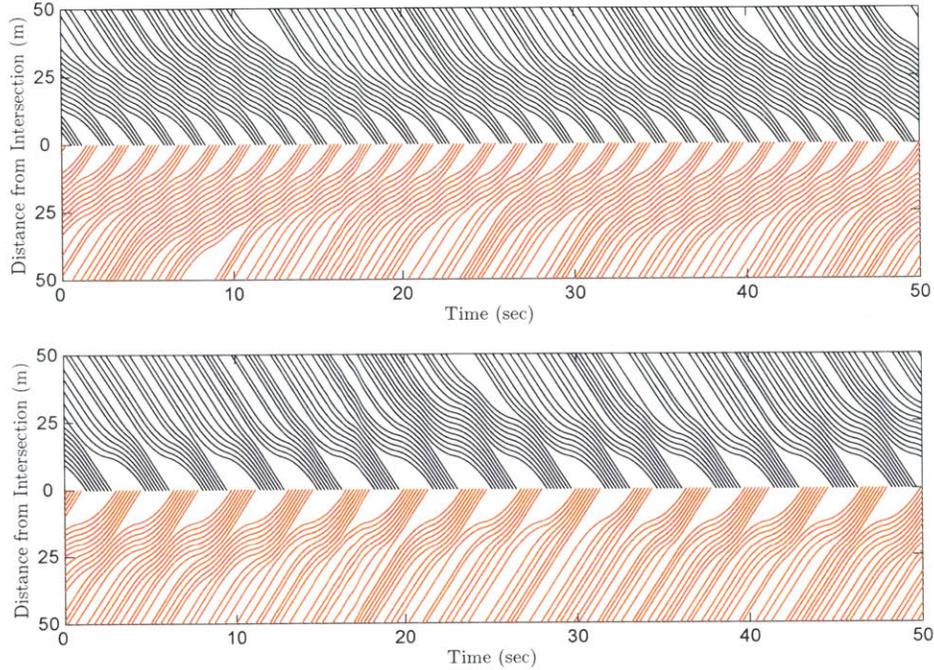


Figure 5-2: The trajectories of the vehicles are shown according to a  $k$ -limited polling policy, where  $k = 4$  in the upper plot and  $k = 8$  in the lower plot.

is in line with Lemma 4.3.1, hence with Theorems 4.3.2 and 4.3.3.

Next, we compute performance bounds on Algorithm 1, assuming the exhaustive policy and vehicles arrive according to a Matérn process. Define the Poisson process with parameter  $\lambda$  as the Poisson point process in the line that has intensity  $\lambda$ . Define the Matérn process with parameter  $\lambda$  as the Matérn process obtained by thinning a Poisson process with parameter  $\lambda$ . Note that the intensity of the Matérn process with parameter  $\lambda$  is

$$\lambda_I = \frac{1 - \exp(-2\lambda b)}{2b},$$

where  $b$  is the service time [27]. In Figure 5-3, we compare the performance of our proposed algorithm with the performance of the corresponding polling systems, which is in line with our result in Theorem 4.3.3. Note that for any  $\lambda$ , we can bound the average delay incurred in Algorithm 1 by the average wait time of an exhaustive polling system with Poisson arrivals of an appropriate intensity.

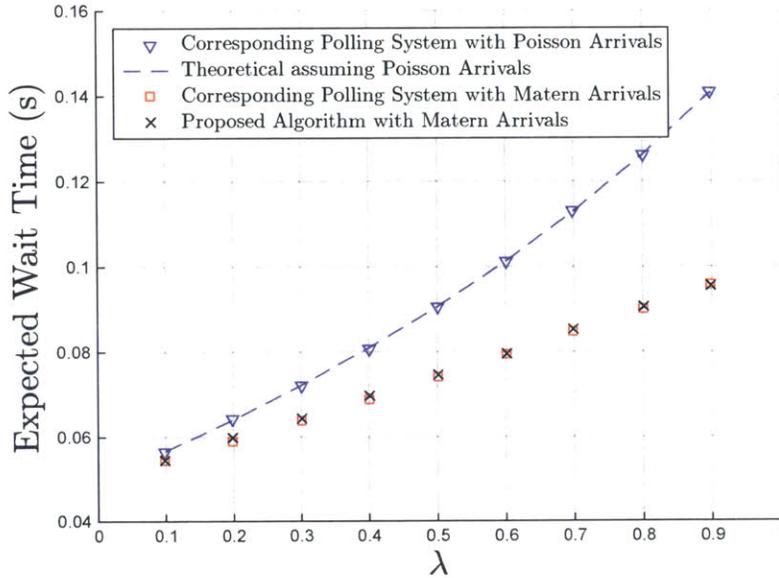


Figure 5-3: The comparison of the performance of the proposed algorithm with the performance of the corresponding polling system.

## 5.2 Overcrowding Assumption

Recall that the manner in which we modeled overcrowding in our system was by thinning out the incoming vehicle before it ever entered the control region if no feasible trajectory existed for it. Although one may argue that the choice of polling policy affects the thinning of incoming vehicles—indeed for a fixed control region length and fixed vehicle arrival rates for each lane, the exhaustive policy, on average, tends to thin less vehicles than the  $k$ -limited policy—because we are considering stochastic arrivals, any algorithm, even one which most optimally motion plans vehicles so as to eliminate thinning, will inevitably require a similar overcrowding assumption. In future work, we hope to relax the overcrowding assumption and consider a more natural model, such as vehicles slowing down before reaching the control region as they approach this congested region.

For the remainder of the simulations carried out in this section, we employed the exhaustive policy, although any regular polling policy can be used and should show similar results. First, let us consider how often a vehicle may be thinned. In Figure 5-4, the percentage of thinned vehicles is plotted against arrival process intensity. These quantities are each per lane, *i.e.*, arrival process intensity for each lane, and percentage of vehicles

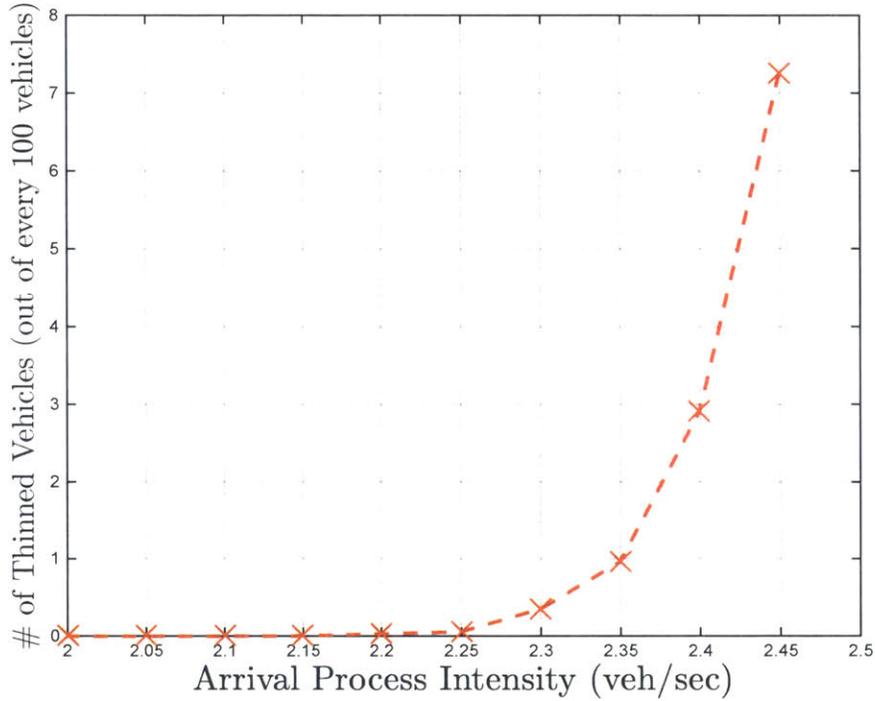


Figure 5-4: Percentage of thinned vehicles as a function of arrival process intensity. The length of the control region is set to  $L^* = 50$  meters.

per lane. We see that until approximately 2.15 vehicles per second, virtually no thinning occurs. These results were obtained by running our proposed algorithm for 50,000 seconds of simulation time, and comparing the number of thinned vehicles per lane to the total number of vehicles that approached that lane. Note that this simulation uses the shortest possible control region length  $L^*$ , as determined by Assumption 4.2.3. For longer control regions, the onset of thinning is pushed even closer to the point of instability (2.5 vehicles per second).

Next, we consider how often the event of a thinned vehicle occurs as a function of control region length. In Figure 5-5, we plot the log of the intensity of thinned vehicles as a function of the control region length. We fixed an arrival rate per lane of 2.45 vehicles per second. Each data point represents 50,000 to 100,000 seconds of simulation time, depending on how quickly the intensity converged. Note that the log of the intensity is approximately linear with road length, or that the number of thinned vehicles decreases exponentially with increasing road length. This suggests that the number of thinned vehicles

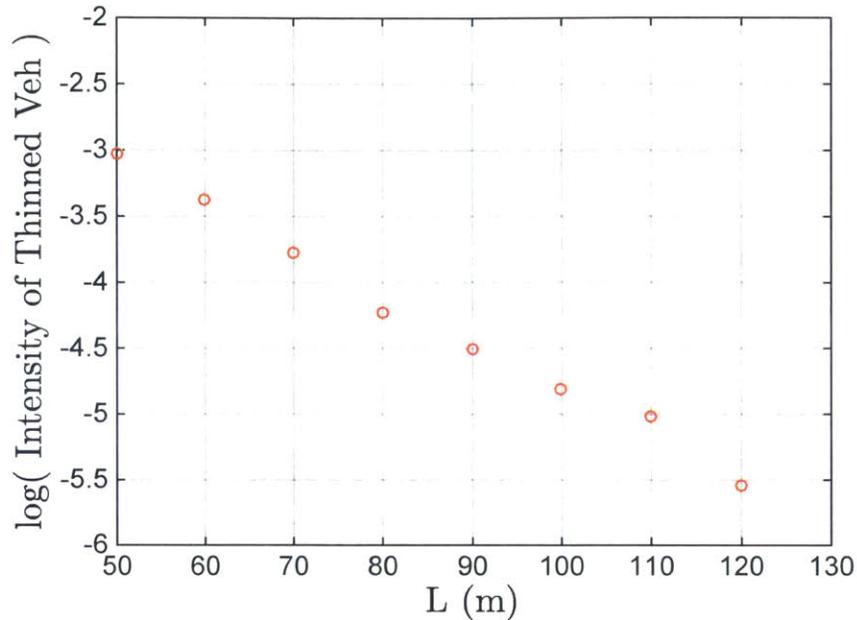


Figure 5-5: Log of the intensity of thinned vehicles, in units of  $\log(\text{veh/sec})$ , is plotted against control region length  $L$ .

is connected with the time that the queue length spends above a certain threshold.

### 5.3 Traffic Light Comparison

In this section, we compare the performance of the proposed algorithm with a traffic light scenario. Two lanes of vehicles are approaching an intersection of same dimensions as in the problem definition in Chapter 2. The traffic light cycles through three phases for each lane in the following order: green, yellow, red, yellow. In the green phase, vehicles quickly accelerate to maximum speed, attempting to pass through the intersection. During the red phase, vehicles decelerate quickly to a full stop, approaching as close to the intersection as safely possible. In the yellow phase, vehicles do one of two things. If the vehicle can come to a full stop before entering the intersection, the vehicle decelerates to a stop as close to the intersection as safely possible; otherwise, the vehicle accelerates quickly and passes through the intersection. This implies that a yellow phase which follows a red phase is simply an extension of the previous red phase. Also, the traffic cycle is staggered so that while one lane is in its green phase, the other is in its red phase, and *vice versa*. Although

both lanes are in the yellow phase simultaneously, due to the staggering of the green and red phases, only one lane permits its vehicles to traverse the intersection, provided that they cannot safely come to a stop before entering the intersection region.

The duration of the yellow phase is a constant depending on the geometry and dynamics of the vehicles, calculated such that it is the minimum amount of time to guarantee safety between the two lanes of intersecting traffic: a vehicle, that cannot decelerate to a stop before the intersection, is able to completely clear the intersection region by the end of that yellow phase. For example, suppose a vehicle is traveling at full speed  $v_m$  at the instant the traffic light switched from the green phase to yellow phase. The distance required for the vehicle to come to a full stop is  $v_m^2/(2a_m)$ . However, suppose the remaining road length in front of the vehicle is slightly less than this quantity, and thus, the vehicle must continue at full speed and clear the intersection. The time to reach and exit the intersection region is slightly less than  $v_m/(2a_m) + (l+w)/v_m$ . To ensure safety, the duration of the yellow phase must be at least as long. One can show that for any other initial configuration of position and velocity at the beginning of the yellow phase, that the time for such a problematic vehicle to clear the intersection is also no greater than  $v_m/(2a_m) + (l+w)/v_m$ . Thus, the duration of the yellow phase is set to this time length.

In the traffic light simulation, time is discretized, and vehicles are populated according to a Matérn process. These vehicles are initialized to maximum velocity, and at each time step, a control action is determined for each vehicle. To keep in line with a traditional traffic light scenario, we assume that in determining its control action, the vehicle can see what phase the lane is currently in, no matter how far away from the intersection the vehicle is located; however, the vehicle does not know how much longer the current phase will last. Also, we assume that the drivers have perfect knowledge of the instantaneous velocity of the vehicle directly in front and also of the distance between the two vehicles. Although the vehicle does not know the future trajectory of the vehicle in front, the vehicle does know what the instantaneous acceleration of the vehicle in front is, *i.e.*, the vehicle knows what the vehicle in front is about to do. Given these constraints, we design control strategies for the vehicles such that they position themselves as close to the vehicle in front as possible, while also preserving safety. The manner in which we do this is by finding, at each time

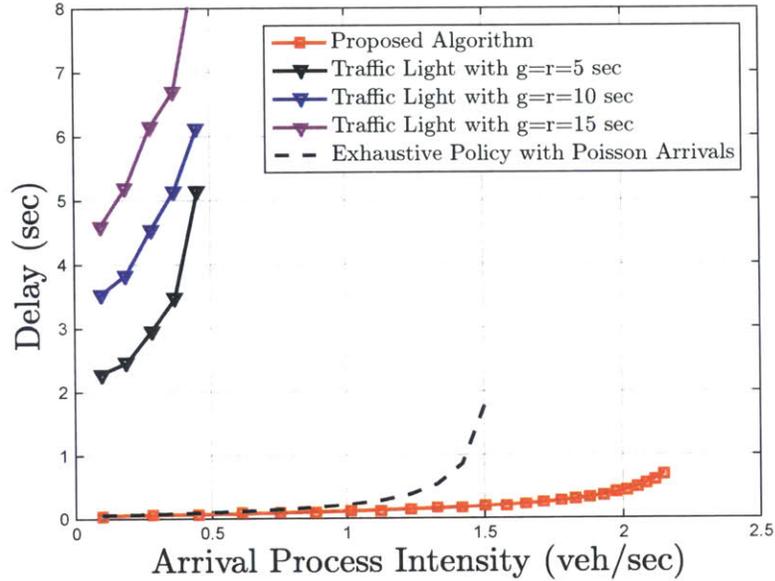


Figure 5-6: Proposed algorithm expected wait time is compared with a traditional traffic simulation with green and red light phases of 5 seconds, 10 seconds, and 15 seconds.

step, the maximum acceleration possible to govern the motion of the vehicle over the next time interval, that will guarantee the following condition is satisfied at the next time step:  $v_j^2/(2a_m) \leq \xi - l + v_{j-1}^2/(2a_m)$ , where  $\xi$  is the distance between the bumpers of the two vehicles, and  $v_j$  and  $v_{j-1}$  are the velocities of the vehicle and the vehicle directly in front, respectively. One can interpret this condition as follows: if the vehicle in front, beginning at the next time step, decelerates with maximum deceleration to a full stop, the vehicle behind can also safely come to a stop. This traffic light intersection is populated by vehicles with aggressive control strategies, and thus, provides a lower bound on the expected wait time of a traditional intersection. Even so note in Figure 5-6, that Algorithm 1 with the exhaustive policy outperforms, in terms of minimizing delay, by at least two orders of magnitude this aggressive traffic light scenario.



# Chapter 6

## Conclusion

In this thesis, we considered the problem of coordinating the motion of vehicles through an intersection with no traffic lights. We proposed a coordination algorithm that provides provable guarantees on both safety and performance in all-autonomous traffic intersections. The proposed algorithm, at its core, schedules vehicles to use the intersection region according to a polling policy, which can be selected from a wide variety of policies. Then, we show that one can motion plan the vehicles in the control region in a safe and high performance fashion. In that regard, this work established novel connections between the polling systems literature and the motion planning, in the context of coordination problems in all-autonomous intersections. Furthermore, provable performance bounds were established for the average delay of the proposed algorithm by considering its corresponding polling policy. Simulations verified these bounds for the case of a Matérn arrival process. A computationally efficient method of approximating the optimal trajectories was established. High performance of the proposed algorithm was compared to a traditional red-yellow-green traffic light populated with aggressive human drivers.

In future work, we plan to make a more realistic overcrowding assumption. For example, suppose that as vehicles approach the congested control region, they slow down, mirroring what happens as cars approach a highly congested traffic intersection. Also, in future work we wish to consider a more complex protocol for handling multi-lane traffic intersections, in which vehicles have the option to continue traveling straight, or turn left or right. This adds significant complexity to the problem. However, a simpler problem that

one can first consider is a multi-lane traffic scenario in which vehicles are only allowed to go straight along their road. We believe that this problem can be solved directly building off the concepts developed in this thesis.

Also, we believe that a minimum road length  $L^*$  exists for broader classes of dynamic models, such as an  $n$ th order integrator with bounded derivatives. This class of dynamic models is rich enough to model accurately many physical systems, since most systems can be sufficiently approximated as a series of first order systems. We believe that a minimum road length can be established for such  $n$ th order integrators with asymmetric bounds on the derivatives. This is quite useful since in general, physical systems such as cars, airplanes, *etc.*, accelerate and decelerate at different rates.

Moreover, this work introduces a new type of polling systems in which customers are subject to differential constraints. We believe that these models may be valuable in a number of applications involving congested environments, including automated warehouses, ports, and factories as well as air traffic control problems.

# Bibliography

- [1] Design of an Automated Transportation System in a Seaport Container Terminal for the Reliability of Operating Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (to appear)*, July 2007.
- [2] Federal Aviation Administration. Amazon gets experimental airworthiness certificate, 2015.
- [3] Amazon.com. Amazon prime air, 2015.
- [4] R Azimi, G Bhatia, R Rajkumar, and P Mudalige. Intersection Management using Vehicular Networks. In *Society for Automotive Engineers (SAE) World Congress*, pages 1–13, January 2012.
- [5] François Baccelli and Paola Bermolen. Extremal versus additive Matérn point processes. *Queueing Systems*, 71(1-2):179–197, March 2012.
- [6] M A A Boon, R D van der Mei, and E M M Winands. Applications of polling systems. *Surveys in Operations Research and Management Science*, 16(2):67–82, July 2011.
- [7] O J Boxma, O Kella, and K M Kosiński. Operations Research Letters. *Operations Research Letters*, 39(6):401–405, November 2011.
- [8] D Carlino, S D Boyles, and P Stone. Auction-based autonomous intersection management. In *IEEE Intelligent Transportation Systems Conference*, September 2013.
- [9] A Colombo and D Del Vecchio. Algorithms for collision avoidance at intersections. In *Hybrid Systems: Computation and Control*, January 2012.
- [10] The Engineer. Amazon uses an army of robot workers in its warehouse to fulfill orders, 2014.
- [11] D Gross and C Harris. *Fundamentals of Queueing Theory*. Wiley, 1998.
- [12] Martin Grunow, Hans-Otto G nther, and Matthias Lehmann. Dispatching multi-load AGVs in highly automated seaport container terminals. *OR Spectrum*, 26(2):211–235, March 2004.
- [13] E Guizzo. Three Engineers, Hundreds of Robots, One Warehouse. *IEEE Spectrum*, pages 26–34, June 2008.

- [14] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.
- [15] M R Hafner, D Cunningham, L Caminiti, and D Del Vecchio. Cooperative Collision Avoidance at Intersections: Algorithms and Experiments. *IEEE Transactions on Intelligent Transportation Systems (to appear)*, March 2013.
- [16] S Hoshino, J Ota, A Shinozaki, and H Hashimoto. Design of an AGV Transportation System by Considering Management Model in an ACT. *Intelligent Autonomous Systems*, January 2006.
- [17] P Stone K Dresner. A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research*, 31:591–656, February 2009.
- [18] R C Larson and A R Odoni. *Urban Operations Research*. MIT Press, 1999.
- [19] H Levy and M Sidi. Polling systems: applications, modeling, and optimization. *IEEE Transactions on Communications*, 38(10):1750–1760, 1990.
- [20] Kai Liu, Edward Chan, Victor Lee, Krasimira Kapitanova, and Sang H Son. Design and Evaluation of Token-based Researvation for a Roadway System. *Transportation Research Part C*, 26:184–202, January 2013.
- [21] M Lupu, E Feron, and Z H Mao. Traffic Complexity of Intersecting Flows of Aircraft Under Variations of Pilot Preferences in Maneuver Choice. In *IEEE Conference on Decision and Control*, October 2010.
- [22] R Lyons. Complexity analysis of the Next Gen Air Traffic Management System: trajectory based operations. *Work A Journal of Prevention, Assessment and Rehabilitation*, 41(1):4514–4522, February 2012.
- [23] Z.-H. Mao, E Feron, and K Bilimoria. Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules. *IEEE Transactions on Intelligent Transportation Systems (to appear)*, 2(2):101–109, June 2001.
- [24] E Onieva, V Milanés, J Villagrà, J Pérez, and J Godoy. Expert Systems with Applications. *Expert Systems With Applications*, 39(18):13148–13157, December 2012.
- [25] Matthew Sparkes. Is apple building a driverless car?, 2015.
- [26] D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic Geometry and Its Applications*. Wiley, 2nd edition, 1995.
- [27] D Stoyan and H Stoyan. On One of Matern’s Hard-core Point Process Models. *Math. Nachr.*, pages 205–214, January 1985.
- [28] H Takagi. Queueing Analysis of Polling Models. *ACM Computing Surveys*, 20(1):5–28, March 1998.
- [29] J Teichmann, F Ballani, and K G van den Boogaart. Generalizations of Matern’s hard-core point processes. *Spatial Statistics*, 3:33–53, September 2012.

- [30] Terex. Automated container transport for performance-orientated terminals, 2015.
- [31] Peter Stone Tsz-Chiu Au. Motion Planning Algorithms for Autonomous Intersection Management. In *Bridging the Gap Between Task and Motion Planning Workshop*, June 2010.
- [32] V M Vishnevskii and O V Semenova. Mathematical methods to study the polling systems. *Automation and Remote Control*, 67(2):173–220, February 2006.
- [33] Wikipedia. Google driverless car, 2015.
- [34] P R Wurman, R DAndrea, and M Mountz. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine*, 29(1):9–19, March 2008.