Polling-Systems-Based Autonomous Vehicle Coordination in Traffic Intersections With No Traffic Signals

David Miculescu and Sertac Karaman

***Abstract*—As autonomous vehicle technology advances rapidly, the design and operation of networks composed of fully autonomous vehicles have attracted immense interest. It is widely anticipated that fully autonomous vehicle networks will drastically improve performance. In this paper, we consider a widely studied problem, in which autonomous vehicles arriving at an intersection adjust their speeds to traverse the intersection as rapidly as possible, while avoiding collisions. We propose a coordination control algorithm, assuming stochastic models for the arrival times of the vehicles. The proposed algorithm extends the widely studied polling systems analysis to the case involving customers subject to second-order differential constraints. We provide provable guarantees on 1) safety, no collisions occur surely, and 2) performance, rigorous bounds on the expected delay. We also provide a stability analysis for the resulting queueing system. We demonstrate the algorithm in an extensive simulation study, providing one to two orders of magnitude improvement in delays over the traditional traffic light.**

***Index Terms*—Agents and autonomous systems, autonomous systems, optimization, queueing systems.**

I. Introduction

AUTONOMOUS vehicles hold the potential to revolutionize transportation and logistics. Self-driving cars may reduce congestion and emissions, while substantially enhancing safety [1], [2]. Drones may be used for delivering goods in urban centers or for providing emergency supplies in disaster-struck areas [3], [4]. Robotic vehicles that shuttle materials to service packaging requests in Amazon warehouses [5], [6] and autonomous trucks that carry cargo containers in seaports [7]–[9] can all benefit from better coordination algorithms to reduce delays and energy consumption, while increasing throughput. Almost all studies on autonomous vehicles, including those mentioned above, envision large *fleets* of autonomous vehicles that work in coordination to enable efficient transportation and logistics services. High-performance coordination algorithms for autonomous vehicles would also pave the way for the design of future urban centers with shared autonomous vehicles [10], next generation air transportation systems [11], and industrial environments that house autonomous vehicles to shuttle goods [12].

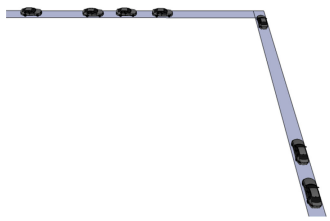


Fig. 1. Illustration of a fully autonomous traffic intersection. Autonomous vehicles arriving near the intersection are controlled by a central control system. The control system ensures that the vehicles safely pass through the intersection; furthermore, the central control system provides provable guarantees on performance, e.g., an upper bound on the average time for a typical vehicle to pass through the intersection.

In systems involving autonomous vehicles, "traffic intersections," where flows of vehicles intersect, are among the most critical for high performance. Efficient coordination algorithms are most beneficial in these places where interaction between vehicles is frequent. In this paper, we study "signalless traffic intersections" in which vehicles coordinate among each other to carefully adjust their speed to maximize traffic performance. Vehicles arrive stochastically at the ends of two roads that intersect at their other end (see Fig. 1). A central controller commands each vehicle, governed by second-order differential constraints with bounded velocity and acceleration. In this setting, we propose a novel coordination algorithm that provides guarantees on safety and performance.

A. Related Work

Most of the existing literature on autonomous intersection control addresses two tightly coupled aspects: i) assign viable time schedules or priorities to vehicles traversing the intersection, and ii) design safe vehicle trajectories fulfilling these assignments. A large body of literature focuses on the second aspect, using heuristics to determine vehicle priority, e.g., first-in-first-out. Trajectory-focused protocols include multi-agent simulations [13]–[18], an auction-based approach [19],

Manuscript received June 12, 2018; revised March 27, 2019; accepted May 4, 2019. Date of publication June 7, 2019; date of current version January 28, 2020. This work was supported by the National Science Foundation under Grant #1350685, Grant #1544413, and an NSF Graduate Research Fellowships Program. Recommended by Associate Editor Dr. U. V. Shanbhag. *(Corresponding author: David Miculescu.)*

The authors are with the Department of Aeronautics and Astronautics, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: dmicul@mit.edu; sertac@mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2019.2921659

discrete-time occupancy trajectories [20], a token-based approach [21], and an MPC-based approach [22]. By using heuristics for aspect (i), these methods focus on safety, often at the expense of optimal traffic performance; for example, the motion planner may revert to a "safe" conservative mode from time to time, to preserve the safety of the intersection, as in [22]. Although these approaches exhibit good performance, a mathematically rigorous analysis of performance, e.g., delay, throughput, and fairness, is not available in these frameworks, which rely on simulation studies to verify performance metrics. In this paper, we utilize well-known results in polling systems theory to determine the (near) optimal crossing order of vehicles in the intersection *in the average case.* In this manner, a rigorous analysis of performance of our algorithm is inherited, as we extend polling systems theory to this intersection problem.

Another group of approaches solve a scheduling problem to address aspect i) informed by the dynamical constraints of the vehicles; and then solve aspect ii) explicitly, based on the solution for aspect (i). In this manner, these methods optimize performance. However, the scheduling problem is computationally challenging; in fact, computing the optimal schedule was shown to be NP-hard [23]. Hence, this second group of literature explores a tradeoff between computational complexity and optimality. In particular, many algorithms use relaxations or approximations, yielding suboptimal solutions or even infeasible solutions at times [24] in order to reduce the computational load of the scheduling problem. The literature in this direction includes Colombo *et al.* [23], [25]–[28], Hult *et al.* [24], and Tallapragada *et al.* [29]. We depart from [23]–[29] in that we focus on optimal performance in the average-case. For instance, while in the optimal control formulation of Hult *et al.* [24] and the scheduling problems considered by many others [23], [25]–[28], the initial states of all vehicles are known *a priori*, our formulation is stochastic and online, i.e., the arrival times of the vehicles are unknown (but statistics are available). The run time of our proposed algorithm scales *linearly* with increasing number of vehicles.

Problems similar to that presented in this paper were also discussed in the context of air traffic control [30]–[42]. Our work is similar in principle to the analysis of two intersecting aircraft flows by Mao *et al.* [39]–[42]. They consider series of aircraft, one after another, aimed at finding the maximum throughput achievable in the worst case by various aircraft maneuvers, such as heading change. In contrast, our analysis is tailored for ground vehicles in a lane, which have to slow down and speed up, but cannot maneuver sideways. Also, our analysis assumes stochastic arrivals, allowing performance results for the average case.

Finally, the algorithm proposed in this paper builds on existing algorithms from polling systems literature [43]–[47]. Motivated by applications in communication systems, transportation systems, and manufacturing, the literature has flourished in the last few decades. Optimal polling policies were characterized for a large class of input processes [48] and analytical expressions were derived for a range of polling policies [43]–[45]. These foundational results have been utilized in several application domains [46], [47], including urban traffic flows [46]. However, to the best of the authors' knowledge, the application of polling systems to all-autonomous traffic intersections is novel. Furthermore, this paper studies for the first time polling systems with differentially constrained customers.

B. Contributions

First, we formulate a stochastic online version of the autonomous intersection problem studied in the literature. This problem generalizes polling systems in two ways: 1) vehicle geometry to model collisions, and 2) vehicle dynamics to model differential constraints. Furthermore, the problem formulation includes regularity assumptions for the proper handling of vehicle arrivals.

Second, we propose a novel online algorithm that provides guarantees on safety, performance, and computational efficiency. Also, we show that the proposed algorithm can emulate a large class of polling policies designed for classical polling systems. Our results indicate that, under reasonable assumptions, the performance analysis of classical polling systems carries over to the autonomous intersection problem considered in this paper. In addition, we show that our algorithm is computationally efficient. In fact, the run time of the proposed algorithm is *linear* in the number of vehicles. To the best of our knowledge, our results bring together the literature on polling systems and the literature on autonomous intersections for the first time.

Finally, we provide further insight into signalless traffic intersection control in the context of *smart platooning*, i.e., effective clustering of vehicles, crucial for achieving high traffic efficiency, as has also been proposed in these high-performing coordination algorithms [29]. To achieve the best throughput, vehicles in the same lane slow down to form a cluster, crossing the intersection as a platoon, minimizing time wasted in switching between lanes. This behavior naturally arises from our motion coordination algorithm.

A preliminary version of our results was presented at the Conference on Decision and Control in 2014 [49]. In addition to the results presented there, this paper provides a full proof of safety and performance results for a slightly more general setting, provides new results on stability, and provides conjectures and open problems based on new simulation results. Also, a distributed version of the algorithm presented here has appeared recently in [50].

C. Organization

This paper is ordered as follows. We formalize the problem of signalless intersection control with stochastic arrivals in Section II. We describe our control policy in Section III. We state our main theoretical results in Section IV, and provide supporting simulation studies in Section V. Concluding remarks are in Section VI. Technical proofs are given in the Appendix.

II. Problem Definition

Consider a traffic intersection in which two orthogonal lanes intersect. Each vehicle is subject to second-order dynamics

$$\ddot{x}(t) = u(t) \tag{1}$$

where $x(t)$ denotes the position of the front bumper of the vehicle, $0 \le \dot{x}(t) \le v_m$ are the velocity constraints, and $|u(t)| \le a_m$ is the maximum acceleration constraint. The region where the two lanes intersect is called the *intersection region*, defined as the set $[0, w] \times [0, w] \in \mathbb{R}^2$. The portion of each lane within a distance L of the intersection region is called the *control region*, defined as the union of two sets: lane 1, $[-L, 0] \times [0, w]$, and lane 2, $[0, w] \times [-L, 0]$. The input signal $u(t)$ of each vehicle is directly controlled by a central control system for all vehicles that are in the control region. We assume that model uncertainties, exogenous disturbances, and communication-related imperfections are neglected.

This central control system does not know *a priori* the precise times that each vehicle will arrive at the control region. We model the arrival times $\{t_{i,k} : i \in \mathbb{N}\}$ as a suitable stochastic process, where $t_{i,k}$ is the time that the ith vehicle from lane $k \in \{1, 2\}$, or vehicle (i, k), enters the control region. We say that a vehicle arrives at the control region at time $t_{i,k}$ when its front bumper is exactly a distance of L away from the intersection region, i.e., $x_{i,k}(t_{i,k}) = -L$, where $x_{i,k}(t)$ denotes the position of vehicle (i, k) at time t. Since then, its position $x_{i,k}(t)$ is governed according to the dynamics given in (1). If the lane number of vehicle (i, k) is clear from context, we drop the lane subscript, i.e., x_i denotes the trajectory of vehicle (i, k), t_i denotes the arrival time of vehicle (i, k), etc. Vehicles enter the control region with maximum speed, i.e., $\dot{x}(t_{i,k}) = v_m$ for all $i \in \mathbb{N}$ and for all $k \in \{1, 2\}$. Since these vehicles are autonomous, we can assume that far away from intersections, the vehicles are designed to travel at the speed limit; however, in future work, we would like to consider a stochastic distribution on the arrival velocities of vehicles. In this more general setting, a motion planner would attempt to increase velocities of the arriving vehicles in order to achieve good performance. We note that the results of this paper can be readily extended to certain scenarios: an intersection with k one-way-traffic lanes intersecting at the origin or in a setting where the lengths of incoming vehicles are governed by a general stochastic distribution. However, generalizing the vehicle dynamics and generalizing the intersection to allow for multiway traffic are both nontrivial extensions of this work. From a practical standpoint, this work would directly be applicable to and efficacious for intersecting highways of traffic.

We represent each vehicle as a two-dimensional, rectangular rigid body with length l and width w. The position of this rigid body, which is the location of its front bumper, is encoded with respect to a corner of the intersection region, see Fig. 2. Define the rigid body at position $y \in \mathbb{R}$ in lane k as $R_k(y) \subseteq \mathbb{R}^2$, i.e., $R_1(y) := (y - l, y) \times (0, w)$ and $R_2(y) := (0, w) \times (y - l, y)$. With this notation, the location of vehicle (i, k) at time t can be represented as $R_k(x_{i,k}(t))$. Let $I_k(t)$ denote the indices for all lane k vehicles inside the control or intersection regions at time t. For instance, if the third, fourth, and fifth 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 $t \ge 0$ and all $k \in \{1, 2\}$. We define safety as follows.

Definition 1 (Safety): The intersection is *safe* at time $t \in \mathbb{R}_{\ge 0}$, if there are no pairwise collisions among the vehicles, i.e., $R_k(x_{i,k}(t)) \cap R_l(x_{j,l}(t)) = \emptyset$ for all $i \in I_k(t)$, all $j \in I_l(t)$ and all $k, l \in \{1, 2\}$ satisfying $(i, k) \ne (j, l)$.

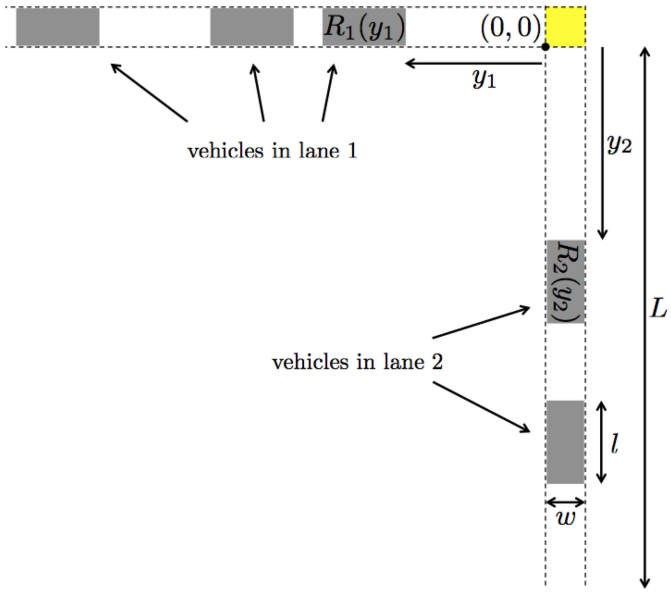


Fig. 2. Illustration of lanes 1 and 2, control region with length L, and the intersection region shaded yellow. Vehicles in the control region are shaded gray, and their positions y_1 and y_2 range in $[-L, 0]$.

Next, for each vehicle, we define the *delay incurred in traveling through the intersection*, or *delay* for short, as follows. Let $T_{i,k}$ be the time that vehicle (i, k) exits the intersection region (the rear bumper is not in the intersection region anymore): $x_{i,k}(T_{i,k}) = l + w$. Then, the time that vehicle (i, k) has spent in the intersection is $T_{i,k} - t_{i,k}$. The delay is the difference between this duration and $(L + l + w)/v_m$, the amount of time required for the vehicle to travel through an empty intersection.

Definition 2 (Delay): The *delay* of vehicle (i, k) is

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

We are interested in developing coordination algorithms such that both performance (e.g., in terms of delay) and safety (avoiding collisions) are simultaneously guaranteed. We discuss such algorithms in the next section. We discuss the arrival process as a part of our model and assumptions in Section IV. It is possible to extend this problem definition to more complex intersections, which we outline in the conclusion.

III. Control Policy

In this section, we propose a coordination algorithm for the intersection problem in Section II. Since our algorithm is based on polling policies that appear in polling systems literature [45], we first briefly introduce queueing and polling systems.

A. Introduction to Polling Systems

A typical queueing model is the following. Suppose *customers* arrive stochastically into a single *queue* in which their requests are processed by a *server*, one by one. The time

required for the server to process a typical customer is called the *service time*. Polling systems are extensions of queueing systems. Suppose customers arrive stochastically into multiple queues. A server chooses to service a customer in any queue. However, the server must pay a cost, called the *switchover time* or the *setup time*, to begin servicing customers in a different queue. No setup time is required, if the server continues to service customers from the same queue.

Consider a polling system with two queues and a single server. Let $t_{i,k}$ denote the arrival time of the ith customer in queue k, and let $s_{i,k}$ denote the service time of this customer, where $k \in \{1, 2\}$. Both $\{t_{i,k} : i \in \mathbb{N}\}$ and $\{s_{i,k} : i \in \mathbb{N}\}$ are stochastic processes. Denote $W_{i,k}$ as the waiting time of the customer, which is the time spent in the queue plus its service time. Customers are serviced by the server one at a time. To switch between queues, the server pays a setup time r, which is a random variable.

Central to a polling system is a controller determining when to switch between queues. The polling systems literature analyzes the performance of various polling policies [45]. Popular examples include the *exhaustive policy* and the *k-limited policy*. In the *exhaustive policy*, the server services customers from one queue until empty. In the *k-limited policy*, the server services at most k customers from one queue per visit. The server visits all queues usually in a cyclic manner. These policies can be formalized. For details, we refer the reader to the seminal paper by Takagi [45].

Most of the existing literature focuses on independent identical memoryless distribution for customer interarrival times. For example, conditions for stability, the mean waiting time, and the mean queue length are known for the three policies considered above [45]. For more general arrival processes, some performance bounds and stability results have been derived for exhaustive and gated policies [51]–[53].

B. Simulating Polling Systems

The coordination algorithm we present in Section III-C relies on simulating the future behavior of a polling system which we describe in this section.

Consider a polling system $\mathcal{P}$ with two queues and deterministic service time s and deterministic setup time r. When a vehicle arrives in lane k of the control region, we add a corresponding customer to the end of queue k of polling system $\mathcal{P}$, via the procedure $\mathcal{P}.\texttt{AddToQueue}(k)$.

We will speak of a vehicle as being in the polling system, when we are referring to its corresponding customer in the polling system. The procedure $\mathcal{P}.\texttt{Simulate}()$ simulates the polling system with the vehicles currently present in the polling system, returning two sequences of future time instances, $\mathcal{T}_1 = (\tau_{i_1,1}, \tau_{i_2,1}, \ldots, \tau_{i_{n_1},1})$ and $\mathcal{T}_2 = (\tau_{j_1,2}, \tau_{j_2,2}, \ldots, \tau_{j_{n_2},2})$, where $\tau_{i,k}$ is the time that the server is tentatively scheduled to begin servicing vehicle (i, k), assuming no future vehicles arrive. We call $\tau_{i,k}$ the *schedule time* of vehicle (i, k). Sets $\mathcal{T}_1$ and $\mathcal{T}_2$ are uniquely determined by the polling policy, the service time s, the setup time r, the order of vehicles in the two queues, and the time that the server begins servicing the first vehicle. Given $\mathcal{T}_1$ and $\mathcal{T}_2$, we define the corresponding *service order of vehicles* $((i_1, k_1), \ldots, (i_n, k_n))$ as the order in which the server will tentatively service the customers according to $\mathcal{T}_1$ and $\mathcal{T}_2$. See Fig. 3 for an example of computing schedule times and service order under the exhaustive policy.

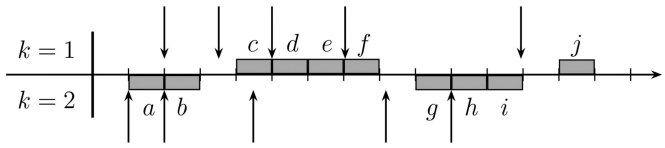


Fig. 3. Vertical arrows depict arrival times of vehicles in lanes 1 and 2. Shaded blocks show the result of simulating the polling system under the exhaustive policy, assuming no other arrivals. Service time s and setup time r are set to $s = r = 1$. Schedule times for lane 1 and lane 2 are $\mathcal{T}_1 = (4, 5, 6, 7, 13)$ and $\mathcal{T}_2 = (1, 2, 9, 10, 11)$. The schedule order is $((a, 2), (b, 2), (c, 1), (d, 1), (e, 1), (f, 1), (g, 2), (h, 2), (i, 2), (j, 1))$.

Algorithm 1: This Algorithm is Triggered When a Vehicle Arrives in the Control Region, Where j is its Lane Number.

$t_0' \leftarrow \texttt{GetCurrentTime}()$;
$\mathcal{P}.\texttt{AddToQueue}(j)$;
$(\mathcal{T}_1, \mathcal{T}_2) \leftarrow \mathcal{P}.\texttt{Simulate}()$;
for $k = 1, 2$ **do**
  for $i = 1, 2, \ldots, n_k$ **do**
    $\tau_{i,k} \leftarrow \mathcal{T}_k[i]$;
    $\mathbf{z_0} \leftarrow (x_{i,k}(t_0'), \dot{x}_{i,k}(t_0'))$;
    $x_{i,k} \leftarrow \texttt{MotionSynthesize}(\mathbf{z_0}, t_0', \tau_{i,k} + \frac{L}{v_m}, x_{i-1,k})$;
  end
end

We emphasize that simulating the polling system does not remove these vehicles from the polling system. A vehicle exits the polling system when the server has completed the service of that vehicle, not when the vehicle receives a tentative schedule time. Moreover, we will see in the next section that a vehicle is designed to be serviced by the server much earlier than the time instant it reaches the intersection region. This implies the existence of vehicles that have exited the polling system, which are still in the control region; the schedule times of such vehicles cannot be altered. Also, the vehicles in the polling system are only a subset of all the vehicles in the control region; such vehicles have schedule times that may be altered upon a later simulation of the polling system.

C. Intersection Coordination Algorithm

We propose an event-triggered algorithm that plans the motions of vehicles in the control region. The algorithm computes trajectory $x_{i,k}$ for vehicle (i, k) in the polling system, such that trajectory $x_{i,k}$ is dynamically feasible and no two vehicles collide. Whenever a new vehicle arrives in the control region, Algorithm 1 is triggered, computing the trajectories $x_{i,1}$ for all $i \in \{i_1, i_2, \ldots, i_{n_1}\}$ and $x_{j,2}$ for all $j \in \{j_1, j_2, \ldots, j_{n_2}\}$, where these index sets correspond only to the vehicles that are currently in the polling system.

Before presenting the coordination algorithm, let us introduce the following motion planner. The procedure

`MotionSynthesize` generates a trajectory for a vehicle given the current time t' of the simulation (not necessarily the vehicle's arrival time), the current state (p, v) of the vehicle, the terminal time t'_f (the time that the vehicle is tentatively scheduled to reach the intersection region, which is not the same as its schedule time), and the trajectory $y : [t_0, t_f] \to \mathbb{R}$ of the vehicle ahead in the same lane. `MotionSynthesize` computes a trajectory $x : [t', t'_f] \to \mathbb{R}$ such that the vehicle reaches the intersection region with full speed v_m at time t'_f, and is safe with trajectory y:

$$\begin{aligned}
&\texttt{MotionSynthesize}((p, v), t', t'_f, y) \\
&\quad := \underset{x:[t',t'_f]\to\mathbb{R}}{\arg\min} \int_{t'}^{t'_f} |x(t)|dt \\
&\qquad \text{subject to } |\ddot{x}(t)| \le a_m \quad \text{for all } t \in [t', t'_f] \\
&\qquad\qquad 0 \le \dot{x}(t) \le v_m \quad \text{for all } t \in [t', t'_f] \\
&\qquad\qquad |x(t) - y(t)| \ge l \quad \text{for all } t \in [t', t_f] \\
&\qquad\qquad x(t') = p; \quad \dot{x}(t') = v \\
&\qquad\qquad x(t'_f) = 0; \quad \dot{x}(t'_f) = v_m .
\end{aligned}$$

The integrand of the cost function is selected such that the vehicle stays as close to the intersection region as possible.[1]

Suppose a vehicle arrives in the control region in lane q, triggering Algorithm 1. The procedure `GetCurrentTime()` returns the current time of the simulation, which is the arrival time of the vehicle triggering Algorithm 1 (Line 1). Next, $\mathcal{P}$.`AddToQueue`(q) adds a customer to the end of queue q, representing the newly arrived vehicle (Line 2). The current polling system is simulated, assuming no future customers, with service time $s = l/v_m$ and setup time $r = w/v_m$ (Line 3). The terminal time of each vehicle (i, k) in the polling system is updated to $\tau_{i,k} + L/v_m$.[2] Finally, `MotionSynthesize` generates a trajectory for each vehicle in the polling system, starting with the first vehicle in each queue and then using this newly generated trajectory as the "trajectory y of the vehicle ahead" when calling `MotionSynthesize` for the second customer, and so on (Lines 4–10).[3]

The durations of the service time and setup time are selected to ensure that no collisions occur in the intersection region. A natural interpretation of the service time is depicted in the transition from Fig. 4(a) to (b), which is the amount of time from when a vehicle's *front* bumper exits the control region to when its *rear* bumper exits the control region, assuming the vehicle travels at speed v_m. The service time of the polling system is set to this duration in order for the polling policy to imitate this behavior when tentatively scheduling vehicles. Upon service completion, the server may immediately begin the service of another vehicle from the *same* queue; similarly, when a vehicle, having reached the intersection region, exits fully the control region, a vehicle from the *same* lane may also safely follow into the intersection region.

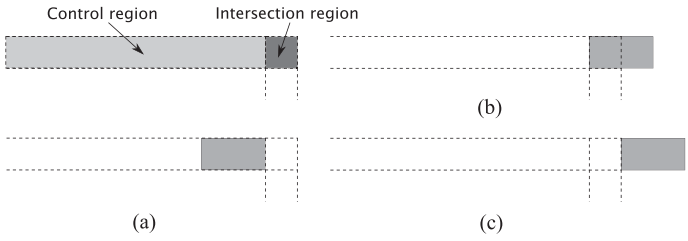


Fig. 4. Control region (shaded light gray) and the intersection region (shaded dark grey) are shown. We show the three time instances that are described in the text. Vehicle motion is to the right. (a) Front bumper of the vehicle is entering the intersection region. (b) Rear bumper is leaving the control region and entering the intersection region. (c) Rear bumper is leaving the intersection region.

The setup time is chosen to match the amount of time needed to ensure that a vehicle from another lane can safely enter the intersection region. This is depicted in the transition from Fig. 4(b) to (c), which is the amount of time from when a vehicle's rear bumper *enters* the intersection region to when its rear bumper *exits* the intersection region.[4] We emphasize that by the design of our algorithm, a vehicle fully exits the control region exactly L/v_m time units *after* the same vehicle had exited the polling system.

IV. Analysis

In this section, we show that the coordination algorithm presented in Section III-C has two important properties. First, the algorithm maintains the safety of the intersection at all times, in that no collisions occur in the control or intersection regions. (See Definition 1.) Second, the algorithm provides good performance in which the delay of a vehicle in the intersection is exactly the waiting time of its corresponding customer in the polling system. These guarantees hold under certain assumptions. We present a class of stochastic process models for vehicle arrival times and two important assumptions. Then, we present our theoretical results.

A. Vehicle Arrival Time Model

We model the arrival times $\{t_{i,k} : i \in \mathbb{N}\}$ as a hard-core stochastic point process on the nonnegative real line [54] such

[1] One may generalize this motion planner to take into account other performance metrics, such as deviation from a prescribed velocity, fuel efficiency, and travel experience. In this paper, we prove the feasibility of our algorithm with the optimal control problem shown here; in future work, we hope to extend the feasibility of the motion planner to a more general setting.

[2] Given schedule time τ, the time scheduled to reach the intersection region is $\tau + L/v_m$. This distinction can be seen in the following scenario. Suppose at time t_0, a vehicle arrives in an empty control region. At time t_0, the vehicle is added to the empty polling system, is given schedule time t_0, and its service begins. At time $t_0 + s$, the server has completed the service and the vehicle exits the polling system, yet the vehicle is still in the control region. Only at time $t_0 + L/v_m$ does the vehicle arrive at the intersection region.

[3] Since the vehicle closest to the intersection region in each lane does not have a vehicle ahead of it, the safety constraint in `MotionSynthesize`, $|x(t) - y(t)| \ge l$, can be ignored when updating its trajectory.

[4] Since the vehicle width does not play a role in our framework other than in determining the setup time, one could interpret it as the width of the intersection region. Similarly, since the vehicle length only plays a role in determining the service time, one could interpret it as a the length of a safe zone around each vehicle.

that $t_{i+1,k} - t_{i,k} \geq l/v_m$ for all $i \in \mathbb{N}$ and $k \in \{1, 2\}$. This inequality ensures that no two vehicles are in collision upon arrival, since the quantity l/v_m is the time required for a vehicle to fully enter the control region. This is a continuous analog of the discrete model for incoming traffic of Stone [13]–[15]. Although our analysis applies to any such processes, in our computational experiments we make use of the widely studied Matérn process, which is generated from a Poisson process [55]–[57]. Poisson processes are commonly used to model traffic arrivals at parking lots, roundabouts, isolated intersections, and networks of intersections [58]–[61]. Poisson processes are considered to be a satisfactory model in the case of light traffic; see [58, Sec. 3.2.1.5] for a discussion of the limitations of the Poisson process in modeling traffic flow.

In addition, we model the effect of overcrowding in the control region to rule out other cases that trivially contradict safety. If a vehicle arrives in a lane at which time no control input would prevent a collision with the vehicle ahead, then the same vehicle does not enter the control region.[5] We call this phenomenon diversion, since this vehicle is *diverted* from the control region. The result is that the stochastic process is "thinned" by the removal of unsafe vehicles. This assumption does not immediately guarantee safety, since the initial schedule time and trajectory of a vehicle are *tentative* and subject to change upon future vehicle arrivals.

However, without this assumption, one is faced with trivial cases in which any algorithm would cause an unsafe intersection with a nonzero probability. In computational experiments, we find that the chance that a vehicle is diverted from the control region is small, when the road length L is very large or the intensity of the arrival times is very small (i.e., in light traffic). We conjecture that our analysis applies in these limiting cases, even without this assumption.

B. Assumptions on Polling Policies and Road Length

We introduce an assumption on the kinds of polling policies that may govern the polling system in the coordination algorithm.

Assumption 1 (Regular Polling Policies): Suppose vehicle A arrives in lane q. At that time, we add vehicle A to the end of queue q and simulate the polling system under a certain polling policy to obtain schedule times $(\mathcal{T}_1', \mathcal{T}_2')$. We say that the polling policy is *regular* if the following two properties hold: 1) vehicles within each queue are serviced in a first-come-first-serve (FCFS) manner, i.e., $\mathcal{T}_k'(i) \leq \mathcal{T}_k'(i+1)$ for all i and all k, and 2) vehicle A is inserted into the previous service order $((i_1, k_1), \ldots, (i_n, k_n))$, without otherwise permuting it, i.e., the updated service order is $((i_l, k_l), \ldots, (i_{m-1}, k_{m-1}), (A, q), (i_m, k_m), \ldots, (i_n, k_n))$, where $1 \leq l \leq m \leq n$.

The FCFS property is a natural requirement, since vehicles within each lane exit the control region in the same order in which they had arrived in that lane. The second property of a regular polling policy implies that a newly arrived vehicle does not influence the service order (and thus, the time schedules) of vehicles present in its lane; and it also implies that the newly arrived vehicle may only delay the schedule times of vehicles in the other queue.

Many widely studied polling policies are regular, as stated in the proposition below, the proof of which is trivial, and thus, omitted.

Proposition 1: The exhaustive, gated, and k-limited polling policies (see Section III-A) satisfy Assumption 1.

Our second assumption is that the length of the control region is bounded from below by a certain distance L^*. This minimum road length L^* is a function of the dynamics of the vehicles, but not of the geometry of the vehicles.[6]

Assumption 2 (The Length of the Control Region): The length L of each of the control regions is bounded from below as follows: $L \geq 2v_m^2/a_m =: L^*$.

As we will show later, Assumptions 1 and 2 allow safe coordination of vehicles while guaranteeing good performance.

C. Theoretical Results: Safety, Performance, and Stability

We first present our theoretical results on the safety and performance guarantees, presented below in Corollaries 1 and 2. These results are enabled by the following lemma.

Lemma 1: Suppose Assumptions 1 and 2 hold. Then, each time a new vehicle arrives and Algorithm 1 is called, every call to the `MotionSynthesize` procedure (Line 8 of Algorithm 1) yields a feasible optimization problem.

Whenever a vehicle arrives in the control region, each vehicle in the polling system is reassigned a terminal time and a trajectory corresponding to its updated terminal time. The lemma states that one can always find a feasible trajectory corresponding to the updated terminal time of a vehicle. The newly arrived vehicle is guaranteed a feasible trajectory due to diversion. Some vehicles have already crossed the intersection region, others are in the intersection but have already exited the polling system; still others are in the polling system, yet are reassigned identical terminal times and trajectories. All these vehicles do not affect the safety of the intersection. Now, certain vehicles in the polling system are reassigned terminal times which have been altered. Only such vehicles may pose a challenge to safety; however, the lemma guarantees that `MotionSynthesize` generates feasible trajectories even for such vehicles. The sketch of this proof can be found in the Appendix.

The following two corollaries follow directly from the lemma. First, the algorithm guarantees safety surely: no collisions occur in the control or intersection regions. Second, the delays of vehicles can be computed from the polling system.

[5]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 (i.e., it is impossible to avoid a collision), then the same vehicle takes the exit at the fork and does *not* enter the control region.

[6]Suppose one envisions remarkably lengthy vehicles; then our results of safety and performance still hold given these assumptions. However, in this case, a significant fraction of vehicles would be diverted, rendering the diversion assumption impractical. In Section V-D, we briefly explore the relationships among diversion, control region length, and arrival rate.

Corollary 1 (Safety): Suppose Assumptions 1 and 2 hold. Then, the intersection is safe at all times $t \geq 0$ in the sense of Definition 1.

Corollary 2 (Performance): Suppose Assumptions 1 and 2 hold. Recall that the delay of vehicle (i,k) is denoted by $D_{i,k}$ (see Section II), and the waiting time of the corresponding customer (see Algorithm 1, Line 2) is denoted by $W_{i,k}$ (see Section III-A). Then, under the coordination algorithm (Section III-C), we have $D_{i,k} = W_{i,k}$, almost surely.

Corollary 2 states that the differential constraints that govern the vehicles are irrelevant. The delay of a vehicle is the waiting time of its corresponding customer in the polling system. Hence, one can employ any (regular) polling policy in the coordination algorithm and inherit its performance. The expected waiting time is known for many polling policies, in particular the exhaustive and gated policies, when the arrival process is Poisson [45].

Our third main result is a stability criterion, presented in Theorem 1. Our notion of a stable intersection is inspired by polling systems literature [45], but adapted for the intersection problem. Let $A_k(t)$ be the number of prediverted vehicles that have "arrived" in lane k during the time interval $[0,t]$. Let $\theta_k^{L,\mathrm{p}}(t)$ be the number of lane k vehicles that are diverted during the time interval $[0,t]$, with parameters control region length L and polling policy p. Consider the following definition of stability.

Definition 3 (Stability): Given the prediverted arrival process $\{t_{i,k}\}$, we say that the coordination algorithm with polling policy p is *stable* if

$$\lim_{L\to\infty}\lim_{t\to\infty}\mathbb{E}\left[\frac{\theta_1^{L,\mathrm{p}}(t)+\theta_2^{L,\mathrm{p}}(t)}{t}\right]=0.$$

If the algorithm is not stable, in the above sense, then we say that the algorithm is *unstable*.

The stability of the coordination algorithm depends on the arrival process and the polling policy, but not on the control region length. Intuitively, stability requires the intensity of the diverted vehicles to vanish as the control region lengthens.

Define $S_k^{L,\mathrm{p}}(t) := A_k(t) - \theta_k^{L,\mathrm{p}}(t)$. The quantity $S_1^{L,\mathrm{p}}(t) + S_2^{L,\mathrm{p}}(t)$ counts the vehicles that have entered the polling system during time interval $[0,t]$, these vehicles can be decomposed into serviced and unserviced vehicles. If the server were to work without rest for a duration of t time units, then the number of serviced vehicles would be bounded above by $1+\frac{t}{s}$. Due to the finite length L of the control region, the number of unserviced vehicles is always bounded above by $2\left(1+\frac{L}{l}\right)$. The inequality $S_1^{L,\mathrm{p}}(t) + S_2^{L,\mathrm{p}}(t) \leq 3 + 2\frac{L}{l} + \frac{t}{s}$ holds. Also, by ergodicity, independence from the initial condition, of the prediverted arrival processes, the equality $\mathbb{E}\left[A_k(t)/t\right] = \lambda_k$ holds for any $t>0$, where λ_k is the intensity of the prediverted arrival process in lane k. We now present a sufficient condition for the instability of the algorithm. The superscript p is dropped for readability.

Theorem 1 (Sufficient Condition for Instability): The coordination algorithm is unstable, if $\lambda_1 + \lambda_2 > \frac{1}{s}$.

Proof: For any $L>0$ and $t>0$, we have

$$\begin{aligned}\theta_1^L(t)+\theta_2^L(t) &= A_1(t)+A_2(t)-\left(S_1^L(t)+S_2^L(t)\right)\\ &\geq A_1(t)+A_2(t)-\left(3+2\frac{L}{l}+\frac{t}{s}\right).\end{aligned}$$

Dividing by t, taking expectation, and then taking the limit, we obtain

$$\lim_{t\to\infty}\mathbb{E}\left[\frac{\theta_1^L(t)+\theta_2^L(t)}{t}\right]=\lambda_1+\lambda_2-\frac{1}{s}>0.$$

Since this holds for any $L>0$, the intersection is unstable. ■

Furthermore, we conjecture the following.

Conjecture 1 (Sufficient Condition for Stability): Let the coordination algorithm have a regular polling policy. If

$$\lambda_1+\lambda_2<\frac{1}{s} \tag{2}$$

then the algorithm is stable.

The restriction to regular polling policies in the conjecture above is crucial. For instance, consider the polling policy that only allows one lane access; this policy is not regular, since vehicles from the neglected lane are never inserted into the service order. Even when arrival rates λ_1 and λ_2 are chosen to satisfy Inequality (2), the intensity of diverted vehicles will always be greater than some $\epsilon>0$, for any L. Thus, this irregular policy creates instability. These conditions are inspired by similar results in the polling systems literature [45]. However, the proof for the sufficient condition for stability in our setting is challenging since thinning the arrival process destroys the stochastic independence assumption between the two lanes, an important assumption made in polling systems literature to derive similar conditions for stability.

V. Computational Experiments

In this section, we evaluate the proposed coordination algorithm in simulation studies. We describe the simulation setup in Section V-A. Then, we describe the results of the computational experiments in light, medium and heavy traffic in Section V-B. Next, we study the effects of diversion in Section V-D. Finally, we compare the performance of our algorithm with an intersection managed by a traditional traffic light in Section V-C.

A. Simulation Environment and Computational Methods

We used MATLAB to generate the simulation results presented here, solving the linear programs (LPs) with Gurobi [62] on a Macbook Pro 2014. Arrival times were generated by a Matérn process [57], independent in each lane, but symmetric, i.e., equal arrival intensities.

The vehicle length, width, minimum velocity, maximum velocity, minimum acceleration, and maximum acceleration were taken to be 2 m, 1 m, 0 m/s, 10 m/s, -4 m/s^2, and 4 m/s^2, respectively. Vehicle trajectories were obtained by a uniform discretization of `MotionSynthesize`. Let Δt be the timestep size, N the number of discretization points, t' the current time in the simulation, and t'_f the terminal time, so

that $\Delta t = (t'_f - t')/(N-1)$. Let p_0 and v_0 be the current position and velocity of the vehicle (at time t'). Let vectors $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{u}$ denote the position, velocity, and control sequences, respectively, with $\mathbf{p}[j]$, $\mathbf{v}[j]$, and $\mathbf{u}[j]$, denoting the jth element of that vector, i.e., $\mathbf{p}[j]$ is the position of the vehicle at time $t' + (j-1)\cdot\Delta t$, etc. Let y be the trajectory of the vehicle ahead. This leads to the following LP:

$$\begin{aligned}
&\arg\max_{\mathbf{p},\mathbf{v},\mathbf{u}} \sum_{j=1}^{N} \mathbf{p}[j] \\
&\mathbf{p}[j+1] = \mathbf{p}[j] + (\mathbf{v}[j] + \mathbf{v}[j+1])\cdot\Delta t/2 \text{ for all } j \\
&\mathbf{v}[j+1] = \mathbf{v}[j] + \mathbf{u}[j]\cdot\Delta t \text{ for all } j \\
&0 \le \mathbf{v}[j] \le v_m \text{ for all } j \\
&-a_m \le \mathbf{u}[j] \le a_m \text{ for all } j \\
&\mathbf{p}[i] \le y(t'_0 + (j-1)\cdot\Delta t) - l \text{ for all } j \\
&\mathbf{p}[1] = p;\ \mathbf{p}[N] = 0;\ \mathbf{v}[1] = v;\ \mathbf{v}[N] = v_m
\end{aligned}$$

where we solve for vectors $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{u}$. To solve this LP for each vehicle, we interpolate the position vector $\mathbf{p}$ of the vehicle ahead when evaluating the (continuous) trajectory y. Also, since positions are in the range $[-L, 0]$, minimizing the sum of absolute values of positions is equivalent to maximizing the sum of positions. The polling system is simulated with service and setup times, as described in Section III-C. We use the "wait-and-see" rule [45] to govern the case of empty queues.[7]

Algorithm 1 with the discretized `MotionSynthesize` is computationally efficient: the run time scales linearly in the number of vehicles, since there at most $2(1+\frac{L}{l})$ calls to the LP upon a vehicle arrival. Also, one can design a hybrid centralized/distributed version of `MotionSynthesize` that is practically implementable [50].

Delay (see Definition 2) was estimated by running the simulation for a long time and averaging the delay of all vehicles. We find that $N = 800$ yields sufficiently smooth trajectories, corresponding to a sample time Δt of about 0.01. In general, the required degree of discretization increases with longer control regions. Given these parameters, we simulated a heavy traffic scenario corresponding to about 2.3 veh/s, which is 91% of the instability limit for the arrival rate, for 100 s of simulation; this took about 22 s of continuous computation time. Similarly, a simulation of 100 s in light traffic, about 1.4 veh/s, which is 57% of the instability limit for arrival rate, took about 7 s of computation time.

B. Effects of Loading Conditions and Platooning

We studied the behavior of our coordination algorithm under various parameters. In Fig. 5, we display vehicle trajectories under the exhaustive policy for low, medium, and high arrival intensities. In Fig. 6, we depict vehicle trajectories under the k-limited policy. During these simulations, `MotionSynthesize` was never infeasible, a consequence of Lemma 1 and Corollaries 1 and 2.

[7]When all queues are empty, the server waits in the queue it last serviced.

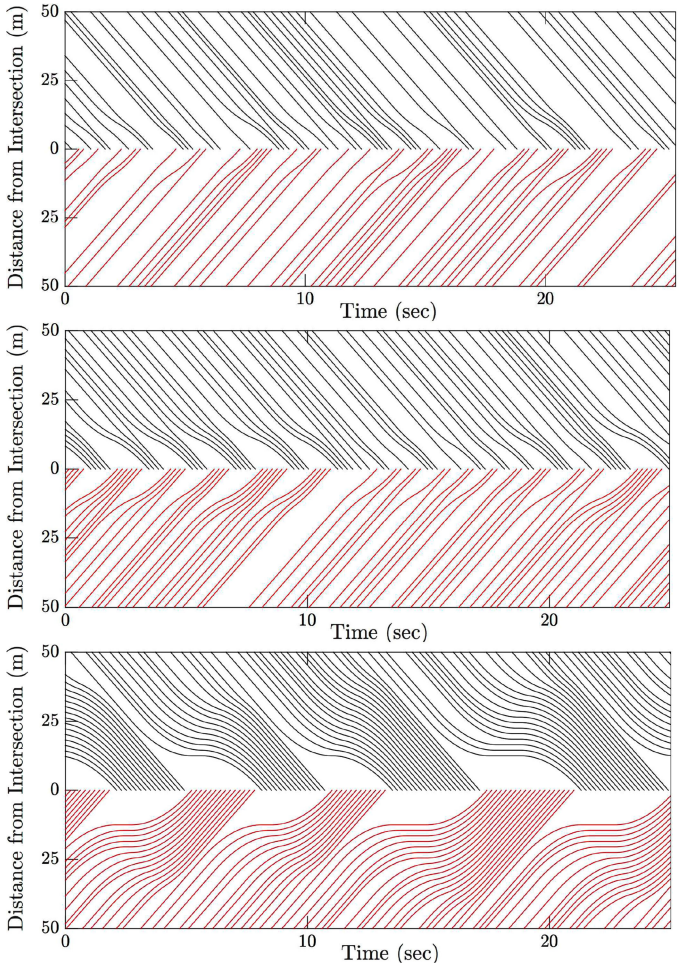


Fig. 5. Vehicle trajectories under (discretized) Algorithm 1 with the exhaustive polling policy. The top, middle, and bottom plots are representative of light, medium, and heavy load traffic, respectively. The upper half of each plot depicts the trajectories of lane 1, while the lower half depicts those of lane 2. Each plot shows a small window of time after the intersection reaches steady state. The average delays from top to bottom are 0.24, 0.53, and 1.6 s, respectively. The arrival rates from top to bottom are 1.7, 2.18, and 2.4 veh/s per lane, respectively.

First, notice that in light traffic, vehicles pass through the intersection with only slight adjustment in speed; low delay is observed in this case. Next, notice the "platooning behavior" in Figs. 5 and 6 when the load becomes substantial (in medium and heavy traffic). Vehicles slow down to form a cluster with other vehicles. This cluster of vehicles speeds up to cross the intersection region at maximum speed. In this way, the intersection region, the shared resource, is utilized as efficiently as possible. Vehicles that slow down do so late in their trajectories, which allows for more vehicles to be in the control region simultaneously.

The performance of our coordination algorithm under the exhaustive policy and Matérn arrivals is plotted in Fig. 7. Define Matérn process with parameter λ as the Matérn process obtained by thinning a Poisson process with intensity λ. The intensity of a Matérn process with parameter λ is $\lambda_I(\lambda) = \frac{1-\exp(-2\lambda s)}{2s}$, where s is the service time [57]. According to Corollary 2, the delay in

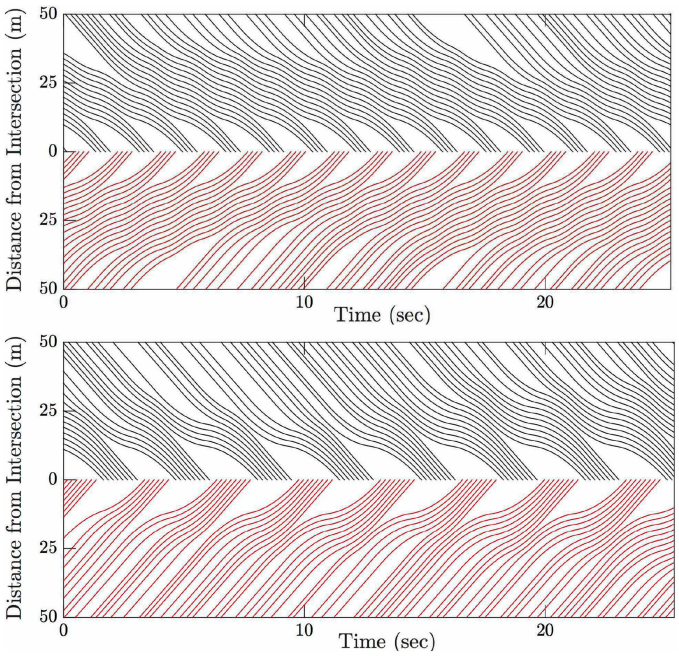


Fig. 6. Vehicle trajectories under (discretized) Algorithm 1 with the k-limited polling policy, $k = 4$ and $k = 8$ in the top and bottom plot, respectively. The upper half of each plot depicts trajectories of lane 1, and the lower half depicts those of lane 2. The plots show the behavior of the intersection after reaching a steady state. The average delays from top to bottom are 2.79 and 1.86 s, respectively. The arrival rate for both the top and bottom plots is 2.3 veh/s per lane.

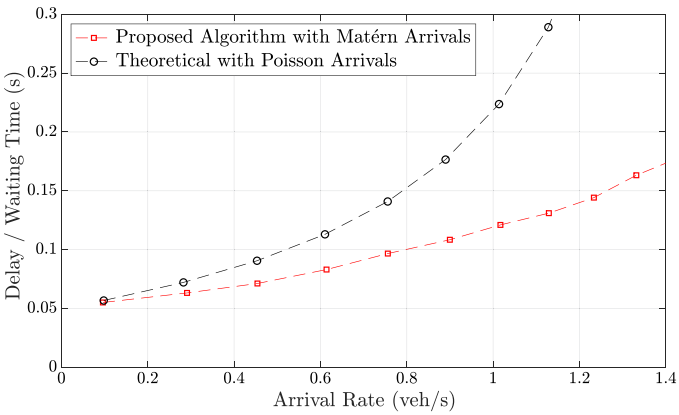


Fig. 7. Comparison of the expected waiting time of a polling system with Poisson arrivals of intensity λ, and of Algorithm 1 according to Matérn arrival process with parameter λ. The exhaustive policy is used.

our algorithm is equal to the waiting time of the polling system with the same arrival process. Also, the expected waiting time of a Matérn arrival process is bounded above by that of a Poisson arrival process.

C. Comparison With a Traditional Traffic Light

We now compare the performance of our coordination algorithm with a discrete-time green-yellow-red phase traffic simulation. This traffic light simulation provides a lower bound on the average delay caused by a traditional traffic light.

Consider two lanes of vehicles approaching an intersection of the same dimensions as described in Section II. Vehicle dynamics and geometry are the same as in Section V-A. However, instead of vehicles arriving (with full velocity) into the control regions according to a Matèrn process *in time*, we populate the control regions with vehicles (at full velocity) according to a Matérn process *in space*, maintaining the same intensity of vehicles.[8] Each vehicle is allowed to maneuver at any point along its lane, no matter its distance to the intersection region. We define delay in the traffic light simulation to be comparable to the definition of delay in Definition 2: the delay of a vehicle is the additional time required for the vehicle to traverse the intersection due to the presence of other vehicles on the road and the traffic light. We compute the average delay by averaging the delays of all vehicles in the traffic simulation over a long period of time.

Each lane cycles through the phases in the following order: green, yellow, red, yellow, and so on, such that one lane is in the green phase while the other lane is in the red phase. The durations of the red and green phases are equal. The duration of the yellow phase is a constant, determined by the vehicle geometry and dynamics: the minimum time required to guarantee safety between two lanes of intersecting traffic.[9]

Vehicles maneuver as follows. The first vehicle in each lane takes into account its lane's phase, while all other vehicles, regardless of the lane phase, maneuver as quickly as possible to catch up and form a platoon with the vehicles ahead. Safety is always maintained, while allowing vehicles to cluster bumper-to-bumper, just as in our coordination algorithm. Furthermore, vehicles move with the maximally allowable acceleration until reaching maximum speed, at which point they continue with this speed.

During a lane's green phase, the first vehicle in that lane accelerates with maximum acceleration, attempting to pass through the intersection. When transitioning from green to yellow phase, the first vehicle of that lane determines whether or not it is able to stop at the entrance of the intersection region. If the vehicle foresees that it cannot, then it accelerates with maximum acceleration and passes through the intersection region. During a lane's red phase, the first vehicle in that lane decelerates with maximum deceleration to come to a stop at the entrance of the

[8]Long traffic light simulation requires a long control region. However, a longer control region does not necessarily incur greater delay, since the definition of delay in the traffic light simulation takes into account the control region length. This adjustment is necessary since, unlike our coordination algorithm, a short control region causes a significantly high fraction of diverted vehicles in the traffic light simulation. On the other hand, the comparison study may become unfair when diversion is significant in our coordination algorithm: in the regime of very high arrival rates, see Fig. 10. Yet even at a high arrival rate of 2.1 veh/s (84% of the instability limit), the fraction of diverted vehicles is only about $10^{-4.6} \approx 2.5 \times 10^{-5}$.

[9]A yellow-phase duration of $\frac{v_m}{2a_m} + \frac{l+w}{v_m}$ is both necessary and sufficient for safety. To derive this quantity, consider the following. Suppose a vehicle is traveling at full speed v_m at the instant its green phase ends. The distance required for the vehicle to fully stop is $v_m^2/(2a_m)$. Suppose the road length ahead is slightly less than this quantity so that the vehicle must continue forward. Then, the time required to pass through the intersection is slightly less than $\frac{v_m}{2a_m} + \frac{l+w}{v_m}$. Thus, the yellow phase must be at least as long. Furthermore, one can show that this duration is also sufficient.

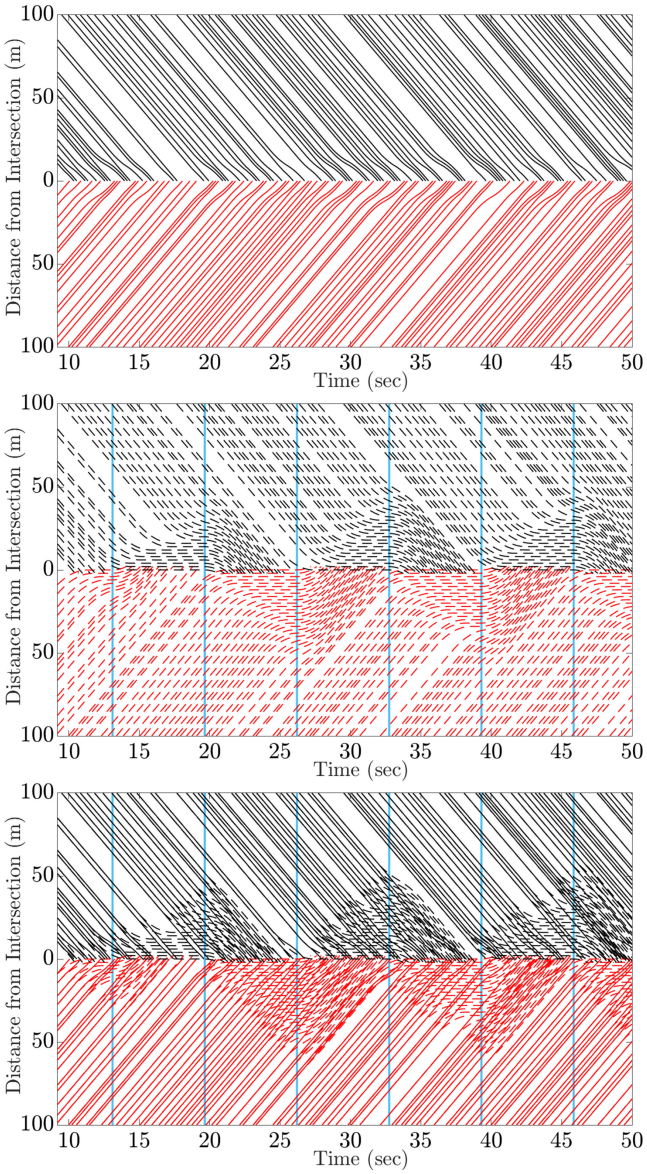


Fig. 8. Arrival intensity is 1.99 veh/s per lane. The top subplot depicts the vehicle trajectories of our proposed algorithm with the exhaustive policy. The middle subplot depicts vehicle trajectories from the traffic light simulation. Identical vehicle arrivals are used as inputs. Blue lines demarcate time instants when either lane 1 or lane 2 enter into the green phase. The bottom subplot superimposes the trajectories of the top and middle subplots.

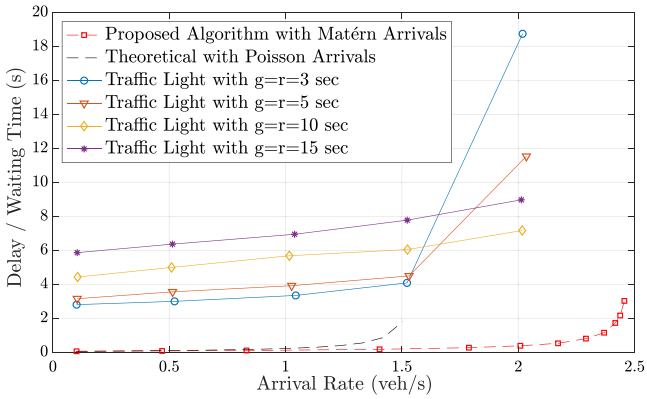


Fig. 9. Delay in intersection governed by Algorithm 1 under the exhaustive policy compared with delay of a traditional traffic intersection with equal green and red light phases of 3, 5, 10, and 15 s. Expected waiting time of a polling system with exhaustive policy and Poisson arrivals is also plotted.

intersection region. A yellow phase immediately following a red phase is simply an extension of the red phase.

In Fig. 8, we compare the vehicle trajectories of our proposed algorithm with those of the traffic light, given identical vehicle arrivals. The intensity of the vehicle arrival stream used is 1.99 veh/s (per lane). The top subplot of Fig. 8 depicts the trajectories of our proposed algorithm under the exhaustive policy. The middle subplot of Fig. 8 depicts trajectories from the traffic light simulation. The green and red phase durations are each 5 s, and a yellow phase duration is used to preserve the safety of the system as described earlier. The blue lines demarcate the time instants when either lane 1 or lane 2 enter into a green phase. The red-dashed trajectories that extend above the 0 m mark and the black-dashed trajectories that extend below the 0 m mark reveal that the vehicles do not cross the intersection region at the same speed. The bottom subplot of Fig. 8 depicts the previous trajectories superimposed on each other. The average delay of the proposed algorithm is 0.35 s, while the delay of the traffic light is 7.1 s.

Further results of the comparison study are provided in Fig. 9. Algorithm 1 with the exhaustive policy outperforms the traffic light scenario by one to two orders of magnitude of delay. Although shorter green/red phases incur lower delays, they engender rapidly increasing queue lengths, a phenomenon indicating premature instability in the system. Note that for an arrival rate of 1.5 veh/s in each lane, the delay in the traffic light scenario increases drastically, even though the instability limit is 2.5 veh/s in each lane. Our intersection algorithm can handle arrival rates even up to 2 veh/s with low delay.

We find that the inefficiency of the traffic light stems from its inefficient use of the intersection region. First, vehicles stop at the boundary of the intersection region, such that the first few vehicles of each green phase take longer to cross the intersection region, incurring more delay; whereas under the coordination algorithm, all vehicles pass through the intersection region at maximum velocity. Second, during the yellow phase, neither lane is fully utilizing the intersection region, while vehicles in both lanes continue to steadily approach the intersection region. The setup time w/v_m of our coordination algorithm may be compared functionally to the yellow phase duration, yet it is an order of magnitude smaller. Third, for long green/red phases, there is roughly a 0.5 chance that a vehicle will arrive during a red phase, incurring significant delay, while for short green/red phases, the inefficient yellow phase becomes a significant part of the total traffic light cycle.

D. Quantifying Diversion

We studied the effect of control region length and arrival rate on diversion. The Matérn arrival process is used along with the exhaustive policy. In our simulations, we noticed that the fraction/intensity of diverted vehicles vanishes exponentially fast

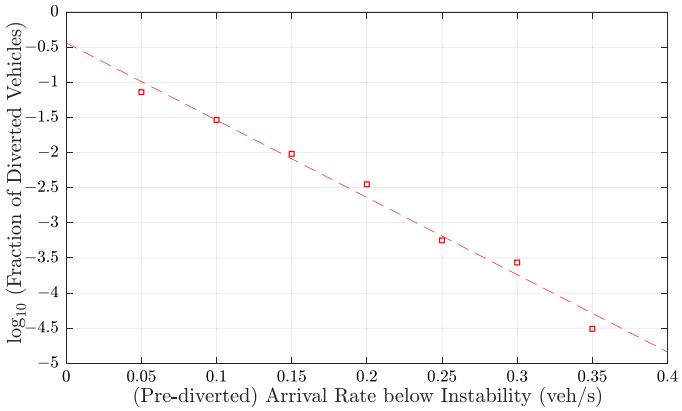


Fig. 10. Logarithm of the fraction of diverted vehicles is plotted against arrival rate below instability. Algorithm 1 is simulated with exhaustive polling policy. The length of the control region is set to the minimum control region length L^* (see Assumption 2) to guarantee safety of Algorithm 1.

with increasing control region length as well as with decreasing (prediverted) arrival rate. In fact, we observe that diversion is extremely rare, when the control region length is reasonably large (e.g., twice the limit given in Assumption 2) and the arrival process is slightly below the instability limit (e.g., 90% of the instability limit).

In Fig. 10, the logarithm of the fraction of diverted vehicles is plotted against the (prediverted) arrival rate below instability. These quantities are each per lane, i.e., arrival rate in each lane, and fraction of vehicles per lane. Up to approximately 2.15 vehicles per second (veh/s), virtually no diversion occurs. These results were obtained from 50 000 s of simulation time. The shortest possible control region length L^* is used (Assumption 2). Longer control regions push the onset of diversion even closer toward the instability limit (2.5 veh/s in each lane).

VI. Conclusion

In this paper, we considered control of vehicles approaching an autonomous intersection. We proposed a coordination algorithm that provides provable guarantees on both safety and performance. The proposed algorithm schedules vehicles to use the intersection region according to a polling policy, which can be selected from a wide variety of policies from the polling systems literature. Provable performance bounds were established for the average delay of the proposed algorithm by considering its corresponding polling policy, and inheriting the rigorous analysis from polling systems theory. Our algorithm is computationally efficient in the sense that the run time scales linearly in the number of vehicles. In simulation studies, the proposed algorithm was compared to a traffic light cycle; it was shown that the proposed algorithm achieves delays of one to two orders of magnitude smaller than a traffic light cycle.

Future work will seek to extend our coordination algorithm to more complex traffic interactions. For example, polling systems with multiple coupled servers [63] seem promising as an analysis tool of complex intersections, where multiple vehicles simultaneously traverse a large intersection region. In this setting, multiple incoming lanes of traffic can be modeled by individual queues and each conflict zone by a server. Furthermore, allowing for a variety of vehicle geometries and terminal velocities can be accommodated by a polling system with random service and setup times. Finally, our approach may be generalized to networks of intersections, using recent analysis of polling system networks [64], [65].

The dynamical aspects of a more complex intersection in the *average case* may be addressed with insights from the worst-case scenario of Mao *et al.* [39]–[42]. A more general class of vehicle dynamics, including sensor impairments and communication delays, may be addressed by lengthening the control region. In all these generalizations, the main challenge is to establish feasibility (Lemma 1) of a coordination algorithm similar to Algorithm 1.

TABLE I
Notation Pertaining to Vehicles at Times t_0 and t_0'

call to Algorithm 1	t_0	t_0'
schedule time	$\tau_{i,k}$	$\tau'_{i,k}$
schedule time of vehicle A	-	τ_A
assigned trajectory	$x_{i,k}$	$x'_{i,k}$
scheduled to arrive at intersection region	$\tau_{i,k} + L/v_m$	$\tau'_{i,k} + L/v_m$

Appendix

We devote the Appendix to the Proof of Lemma 1. We present a number of intermediate results leading to the Proof of Lemma 1. Then, we provide proofs of these intermediate results separately. Due to page limitations, the proofs of the intermediate results are sketched here; the detailed proofs can be found in the full-version available online [66].

(Lemma 1) Suppose Assumptions 1 and 2 hold. Then, each time a new vehicle arrives and Algorithm 1 is called, every call to the `MotionSynthesize` procedure (Line 8 of Algorithm 1) yields a feasible optimization problem.

Suppose a new vehicle, denoted vehicle A, enters the control region at time t_0'. Without loss of generality, we assume vehicle A enters lane 2. By our diversion assumption, there exists a feasible trajectory for vehicle A. Suppose every call to `MotionSynthesize` strictly before time t_0' was feasible. To prove the lemma, we show that at time t_0' each call to `MotionSynthesize` is feasible.

We introduce the following notation. Denote t_0 as the last time Algorithm 1 had been called strictly before the arrival of vehicle A, i.e., $t_0 = \max\{t < t_0' : \text{Algorithm 1 is called at time } t\}$. Denote $\tau_{i,k}$ as the schedule time of vehicle (i,k) that had been assigned at time t_0. Denote $x_{i,k}$ as the trajectory of vehicle (i,k) that had been assigned by `MotionSynthesize` at time t_0. Denote $\tau'_{i,k}$ and $x'_{i,k}$ as the schedule time and trajectory assigned at time t_0'. Denote τ_A as the schedule time of newly arrived vehicle A assigned at time t_0'. See Table I for a summary of this notation.

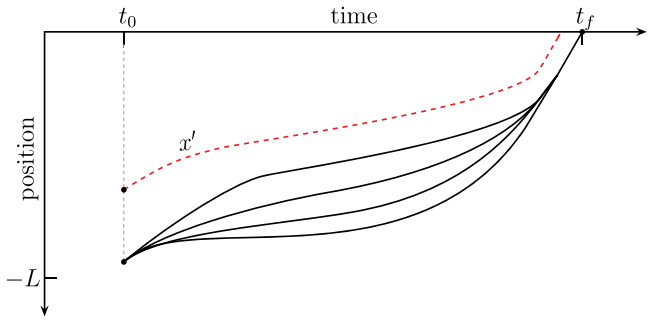


Fig. 11. Sample trajectories from set $\mathscr{C}(\mathbf{z}_0, t_0, t_f, x')$ are shown in black. Trajectory x' is shown as dashed red line. Sample trajectories have the same initial and terminal states, same terminal time, and are safe with x'.

We use the boldface notation as follows. Denote a state $\mathbf{z} := (p, v)$. Denote a state history $z := (x, \dot{x})$. Denote $\mathscr{C}(\mathbf{z}_0, \tilde{t}_0, \tilde{t}_f, x')$ as the set of all trajectories with initial state $\mathbf{z}_0 := (p_0, v_0)$ at time $\tilde{t}_0$ that reach the intersection region with full velocity at time $\tilde{t}_f$ and are safe with some trajectory x'; see Fig. 11. More precisely, define the *set of feasible trajectories* as

$$\begin{aligned}\mathscr{C}(\mathbf{z}_0, \tilde{t}_0, \tilde{t}_f, x') := \big\{ x : [\tilde{t}_0, \tilde{t}_f] \to \mathbb{R} \;\big|\; \exists u \in \mathcal{U} \\ \ddot{x}(t) = u(t),\ |u(t)| \le a_m \quad \forall t \in [\tilde{t}_0, \tilde{t}_f] \\ \dot{x}(t) \in [0, v_m] \quad \forall t \in [\tilde{t}_0, \tilde{t}_f] \\ z(\tilde{t}_0) = \mathbf{z}_0;\ z(\tilde{t}_f) = (0, v_m) \\ |x(t)| \ge l + |x'(t)| \quad \forall t \in [\tilde{t}_0, \tilde{t}_f] \cap \mathscr{D}(x') \big\}\end{aligned}$$

where $u \in \mathcal{U}$ is a measurable function, $u : [\tilde{t}_0, \tilde{t}_f] \to \mathbb{R}$, and $\mathscr{D}(x')$ is the domain of trajectory x'. Also, we write $\mathscr{C}'_{i,k}$ for $\mathscr{C}(z_{i,k}(t'_0), t'_0, \tau'_{i,k} + L/v_m, x'_{i-1,k})$. The call to `MotionSynthesize` at time t'_0 for vehicle (i,k) computes

$$x'_{i,k} \leftarrow \arg\min_{x \in \mathscr{C}'_{i,k}} \int_{t'_0}^{\tau'_{i,k} + L/v_m} |x(t)| dt. \tag{3}$$

Hence to prove Lemma 1, one must check that (3) is feasible for each vehicle (i,k) present in the polling system at time t'_0. We restate Lemma 1 using the new notation.

Lemma 2: Suppose Assumptions 1 and 2 hold. Suppose a new vehicle arrives at time t'_0 and Algorithm 1 is called. Then, for every vehicle (i,k) in the polling system at time t'_0, the optimization problem (3) is feasible.

To prove this lemma, we consider two cases separately: 1) vehicles that are now scheduled to arrive at the intersection region before vehicle A, and 2) vehicles that are now scheduled to arrive at the intersection region after vehicle A.

Lemma 3: Suppose that at time t'_0, vehicle (i,k) is scheduled before vehicle A, i.e., $\tau'_{i,k} < \tau_A$. Then, $\mathscr{C}'_{i,k}$ is nonempty and the minimum in (3) is attained. Moreover, the trajectory $x'_{i,k}$ is simply a truncation of $x_{i,k}$, i.e., $x'_{i,k} = x_{i,k}|_{\{t : t \ge t'_0\}}$.

This lemma states that vehicles now scheduled to arrive at the intersection region before vehicle A simply continue on their previous trajectories. We prove this by induction, using Bellman's optimality principle to show that the previous trajectory is still optimal, in in [66, Sec. A of the Appendix].

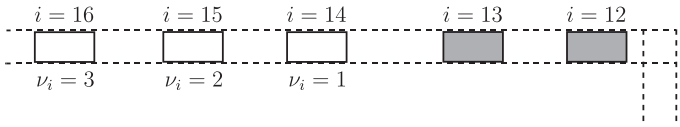


Fig. 12. Lane 1 is shown in full. Vehicles scheduled before vehicle A are shaded gray. Vehicles scheduled after vehicle A have white interiors. For each vehicle scheduled after vehicle A, its value of ν_i is displayed directly below.

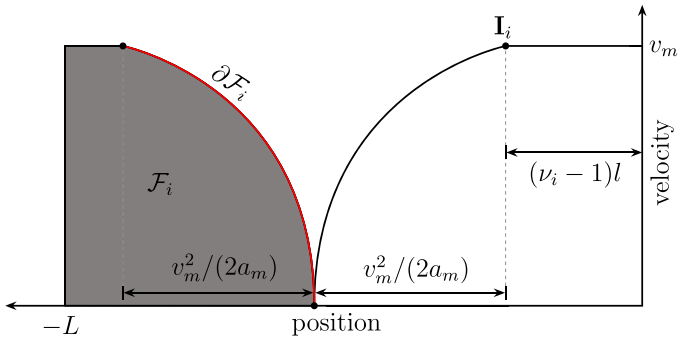


Fig. 13. Sets $\mathcal{F}_i$ and $\partial\mathcal{F}_i$ are shown. Set $\mathcal{F}_i$ is shaded in gray. Set $\partial\mathcal{F}_i$, the right-hand side boundary of $\mathcal{F}_i$, is traced in red. State $\mathbf{I}_i$ has coordinates $(-(\nu_i - l)l, v_m)$. Set $\mathcal{F}_i$ can be described as the set of all states from which state $\mathbf{I}_i$ can be reached at a time indefinitely far into the future. We use doubly arrowed lines to depict absolute values of lengths.

Next, we verify (3) for vehicles that are now scheduled later than vehicle A. We prove the following lemma.

Lemma 4: Suppose that at time t'_0 vehicle (i,k) is scheduled after vehicle A, i.e., $\tau'_{i,k} > \tau_A$. Then, $\mathscr{C}'_{i,k}$ is nonempty and the minimum in (3) is attained.

Since all regular polling policies (see Assumption 1 for a definition) are *first-come first-serve* (FCFS), vehicles scheduled after vehicle A must be in lane 1, not in the lane of vehicle A. For the remainder of the proof, we drop the k subscript, i.e., denote $\tau'_{i,k}$ by τ'_i, and $x_{i,k}$ by x_i, etc. For each vehicle i scheduled after vehicle A, define $\nu_i := i - \min\{\, i \mid \tau'_{i,1} > \tau_A \,\} + 1$. Vehicle i is the ν_i^{th} vehicle scheduled after vehicle A, see Fig. 12. For each vehicle i scheduled after vehicle A, define

$$\begin{aligned}\mathcal{F}_i := \big\{\, (p_0, v_0) \in [-L, 0] \times [0, v_m] \;\big|\; \\ p_0 + v_0^2/(2a_m) \le -v_m^2/(2a_m) - (\nu_i - 1)l \,\big\}.\end{aligned}$$

Denote the right-hand side boundary of the set $\mathcal{F}_i$ as $\partial\mathcal{F}_i$ (see Fig. 13)

$$\begin{aligned}\partial\mathcal{F}_i := \big\{\, (p_0, v_0) \in [-L, 0] \times [0, v_m] \;\big|\; \\ p_0 + v_0^2/(2a_m) = -v_m^2/(2a_m) - (\nu_i - 1)l \,\big\}.\end{aligned}$$

Let us provide some insights into sets $\partial\mathcal{F}_i$ and $\mathcal{F}_i$. For $\nu_i = 1$, vehicle i, moving along $\partial\mathcal{F}_i$, will fully stop a distance $v_m^2/(2a_m)$ from the intersection region. For $\nu_i > 1$, vehicle i, moving along $\partial\mathcal{F}_i$, will fully stop an additional $\nu_i - 1$ vehicle lengths. Hence, k vehicles with $\nu_i \in \{1, 2, \ldots, k\}$, moving

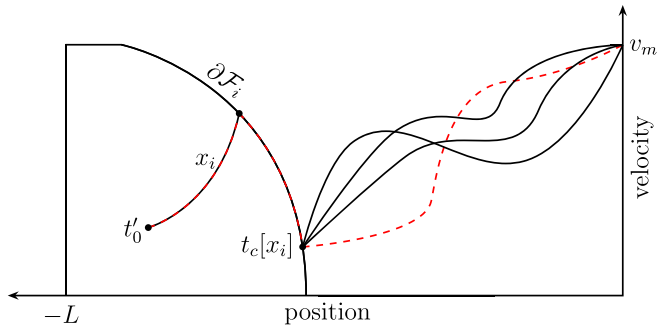


Fig. 14. Trajectory x_i is depicted in position-velocity space as the dashed red line; it begins at time t_0', exits set $\mathcal{F}_i$ at time $t_c[x_i]$, and continues to the state $(0, v_m)$. Sample trajectories from set $\mathscr{E}_i'$ also begin at time t_0', move along trajectory x_i until time $t_c[x_i]$, and then choose any (feasible and safe) path to state $(0, v_m)$. Sample trajectories of $\mathscr{E}_i'$ are shown in black.

along their respective $\partial\mathcal{F}_i$, all fully stop, bumper-to-bumper, such that the foremost vehicle is a distance $v_m^2/(2a_m)$ from the intersection region. These vehicles can move together as a platoon at maximum acceleration (a_m) to reach the intersection with maximum speed (v_m).

Set $\mathcal{F}_i$ is critical in proving Lemma 4. We prove Lemma 4 in three steps. First, we show that the state $z_i(t_0')$ of ν_i^{th} vehicle after vehicle A at time t_0' is in set $\mathcal{F}_i$, see Fig. 13. Next, we show that this vehicle has a *feasible* trajectory satisfying a particular property: the ν_i^{th} vehicle moves along its last assigned trajectory x_i as long as possible (until time $t_c[x_i]$, which we will define later), see Fig. 14. Finally, we show that `MotionSynthesize` returns such a trajectory. This last step is technical, requiring results from optimal control theory to show (3) is well posed. These steps are formalized in Lemmas 5, 6, and 7, respectively. We state these lemmas below, providing insights into their proofs.

First, we show that at time t_0', the state of the ν_i^{th} vehicle after vehicle A is in $\mathcal{F}_i$. These vehicles are able to stop far enough from the intersection region, such that they can attain full speed when crossing it.

Lemma 5: For any vehicle i scheduled after vehicle A at time t_0', i.e., $\tau_i' > \tau_A$, we have $z_i(t_0') \in \mathcal{F}_i$.

Define state $\mathbf{I}_i := (-(\nu_i - 1)l, v_m)$, see Fig. 13. By the construction of `MotionSynthesize`, the ν_i^{th} vehicle after vehicle A arrives at state $\mathbf{I}_i$ and then continues at maximum speed until the intersection region. One can observe this phenomenon in the heavy load case of Fig. 5. Due to the regular polling policy condition (Assumption 1) and the minimum road length condition (Assumption 2), it turns out that the time between the arrival of vehicle A and the arrival of vehicle i at state $\mathbf{I}_i$ (according to previous trajectory x_i) is at least $2v_m/a_m$, i.e.,

$$\tau_i + L/v_m - (\nu_i - 1)s - t_0' \geq 2v_m/a_m.$$

By using Pontryagin's Minimum Principle, we show that $z_i(t_0') \in \mathcal{F}_i$. Note that if $z_i(t_0') \notin \mathcal{F}_i$, then there can be only two outcomes: either vehicle i is not able to reach state $\mathbf{I}_i$, or vehicle i can reach state $\mathbf{I}_i$ but only in an amount of time strictly less than $2v_m/a_m$, violating the above inequality. Detailed proof is given in [66, Sec. B of the Appendix].

Second, we show that not only does there exist a *feasible* trajectory for vehicle i, i.e., $\mathscr{C}_i' \neq \emptyset$, but we also show that vehicle i moves along its last assigned trajectory x_i for as long as possible, see Fig. 14. Denote $\mathscr{E}(y, \tilde{t}_0, \tilde{t}_f, x')$ as the set of all extensions of trajectory y starting at time $\tilde{t}_0$ that reach the intersection region at time $\tilde{t}_f$ and are safe with trajectory x'. More precisely, we define

$$\mathscr{E}(y, \tilde{t}_0, \tilde{t}_f, x') := \big\{x \in \mathscr{C}\big(\big(y(\tilde{t}_0), \dot{y}(\tilde{t}_0)\big), \tilde{t}_0, \tilde{t}_f, x'\big) \;\big|\; x(t) = y(t) \quad \forall t \in [\tilde{t}_0, \tilde{t}_f] \cap \mathscr{D}(y)\big\}.$$

We denote $t_c[x_i]$ as the time that trajectory x_i leaves the set $\partial\mathcal{F}_i$, i.e., $t_c[x_i] := \sup\{t \mid z_i(t) \in \partial\mathcal{F}_i\}$. For shorthand, we write $\mathscr{E}_i'$ for $\mathscr{E}(x_i|_{[t_0, t_c[x_i]]}, t_0', \tau_i' + L/v_m, x_{i-1}')$. Note that $\mathscr{E}_i' \subseteq \mathscr{C}_i'$. We now formalize our second intermediate result.

Lemma 6: Consider vehicle i scheduled after vehicle A at time t_0', i.e., $\tau_i' > \tau_A$. Suppose $z_i(t_0') \in \mathcal{F}_i$ and $x_{i-1}' \in \mathscr{E}_{i-1}'$. Then, $\mathscr{E}_i' \neq \emptyset$.

This lemma states that there always exists a feasible trajectory for the ν_i^{th} vehicle such that it moves along its last assigned trajectory x_i while in set $\mathcal{F}_i$, i.e., until time $t_c[x_i]$, see Fig. 14.

The proof of Lemma 6 is the construction of a trajectory in the set $\mathscr{E}_i'$, given in [66, Sec. C of the Appendix]. This construction is quite elaborate in the general case. However, we present here a simple case that demonstrates the basic building blocks of the construction. Consider the case $\nu_i = 1$. We construct a feasible trajectory in $\mathscr{E}_i'$ for this vehicle as follows: vehicle i moves along its last assigned trajectory x_i until it reaches the boundary $\partial\mathcal{F}_i$, decelerates along $\partial\mathcal{F}_i$, accelerates to full speed, and then continues at full speed. The transition point from deceleration to acceleration hinges on the vehicle's updated terminal time $\tau_i' + L/v_m$. Then, we argue that this constructed trajectory must be safe with the updated trajectory x_{i-1}' of the vehicle ahead. Clearly, the constructed trajectory is safe with x_{i-1}; also, x_{i-1} and x_{i-1}' are identical, by Lemma 3. The general case $(\nu_i > 1)$ is more complicated in its construction due to the fact that the updated trajectory x_{i-1}' of the vehicle ahead changes.

Now, we formalize our final intermediate result.

Lemma 7: Suppose vehicle i is scheduled after vehicle A at time t_0', and $\mathscr{E}_i' \neq \emptyset$. Then, (3) returns a solution from the set $\mathscr{E}_i' \subseteq \mathscr{C}_i'$.

This lemma states that if $\mathscr{E}_i'$ is nonempty, then `MotionSynthesize` admits a trajectory in $\mathscr{E}_i'$. This straightforward lemma is technical in nature, while adding no further insight. The details of the proof of Lemma 7 are given in [66, Sec. D of the Appendix].

Now, let us give a brief overview. From Lemmas 3 and 4, the proof of Lemma 1 follows. Recall that Lemma 3 verifies the feasibility of (3) for vehicles with updated schedule times *earlier* than vehicle A, i.e., $\tau_{i,k}' < \tau_A$. Similarly, Lemma 4 shows the feasibility of (3) for vehicles with updated schedule times *later* than vehicle A, i.e., $\tau_{i,k}' > \tau_A$. We prove Lemma 4 by induction, using Lemmas 5–7.

Consider the case $\nu_i = 1$. By Lemmas 5 and 6, $\mathscr{E}_i'$ is nonempty. Next, by Lemma 7, trajectory x_i', as defined in (3),

exists *and* is contained in $\mathscr{E}_i'$. Now, consider the case $\nu_i > 1$, and assume (by induction) that trajectory x_{i-1}', as defined in (3), exists and is contained in $\mathscr{E}_{i-1}'$. Then, by Lemmas 5 and 6, the set $\mathscr{E}_i'$ is nonempty. By Lemma 7, `MotionSynthesize` admits a solution x_i' contained in $\mathscr{E}_i' \subseteq \mathscr{C}_i'$.

References

[1] J. B. Greenblatt and S. Saxena, "Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles," *Nature Climate Change*, vol. 5, no. 9, pp. 860–863, Jul. 2015.

[2] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transp. Res. Part A*, vol. 77, pp. 167–181, 2015.

[3] K. B. Sandvik and K. Lohne, "The rise of the humanitarian Drone: Giving content to an emerging concept," *J. Int. Stud.*, vol. 43, no. 1, pp. 145–164, Sep. 2014.

[4] R. D'Andrea, "Guest editorial can drones deliver?," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 647–648, Jun. 2014.

[5] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectr.*, vol. 45, no. 7, pp. 26–34, Jun. 2008.

[6] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, pp. 9–19, Mar. 2008.

[7] S. Hoshino and J. Ota, "Design of an automated transportation system in a seaport container terminal for the reliability of operating robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2007, pp. 4259–4264.

[8] S. Hoshino, J. Ota, A. Shinozaki, and H. Hashimoto, "Design of an AGV transportation system by considering management model in an act," in *Proc. 9th Int. Conf. Intell. Auton. Syst.*, 2006, pp. 505–514.

[9] M. Grunow, H.-O. Günther, and M. Lehmann, *Dispatching Multi-Load AGVs in Highly Automated Seaport Container Terminals*. New York, NY, USA: Springer, 2005.

[10] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios," *Transp. Res. C, Emerg. Technol.*, vol. 40, pp. 1–13, 2014.

[11] R. Lyons, "Complexity analysis of the Next Gen Air Traffic Management System: Trajectory based operations," *Work A J. Prevention, Assessment Rehabil.*, vol. 41, no. 1, pp. 4514–4522, Feb. 2012.

[12] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, p. 9, 2008.

[13] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2004, pp. 530–537.

[14] P. S. K Dresner, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Feb. 2009.

[15] P. S. Tsz-Chiu Au, "Motion planning algorithms for autonomous intersection management," in *Proc. 1st AAAI Conf. Bridging Gap Between Task Motion Planning Workshop*, Jun. 2010, pp. 2–9.

[16] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," in *Proc. Soc. Automot. Engineers World Congr.*, Jan. 2012, pp. 1–13.

[17] S. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Reliable intersection protocols using vehicular networks," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2013, pp. 1–10.

[18] R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "Stip: Spatio-temporal intersection protocols for autonomous vehicles," in *Proc. ACM/IEEE 5th Int. Conf. Cyber-Phys. Syst.*, 2014, pp. 1–12.

[19] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2013, pp. 529–534.

[20] Q. Lu and K.-D. Kim, "Intelligent intersection management of autonomous traffic using discrete-time occupancies trajectory," *J. Traffic Logistics Eng. Vol.*, vol. 4, no. 1, pp. 1–6, 2016.

[21] K. Liu, E. Chan, V. Lee, K. Kapitanova, and S. H. Son, "Design and evaluation of token-based reservation for a roadway system," *Transp. Res. C*, vol. 26, pp. 184–202, Jan. 2013.

[22] P. Dai, K. Liu, Q. Zhuge, E. H.-M. Sha, V. C. S. Lee, and S. H. Son, "Quality-of-experience-oriented autonomous intersection control in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1956–1967, Jul. 2016.

[23] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proc. 15th ACM Int. Conf. Hybrid Syst., Comput. Control*, 2012, pp. 145–154.

[24] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *Proc. Amer. Control Conf.*, 2015, pp. 763–768.

[25] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1162–1175, Sep. 2013.

[26] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, vol. 60, no. 6, pp. 1515–1527, Jun. 2015.

[27] G. R. de Campos, F. Della Rossa, and A. Colombo, "Optimal and least restrictive supervisory control: Safety verification methods for human-driven vehicles at traffic intersections," in *Proc. IEEE 54th Annu. Conf. Decis. Control*, 2015, pp. 1707–1712.

[28] A. Colombo, "A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections," in *Proc. 11th Int. Symp. Wireless Commun. Syst.*, 2014, pp. 449–453.

[29] P. Tallapragada and J. Cortés, "Hierarchical-distributed optimized coordination of intersection traffic," *IEEE Trans. Intell. Transp. Syst.*, 2019.

[30] M. Lupu, E. Feron, and Z.-H. Mao, "Traffic complexity of intersecting flows of aircraft under variations of pilot preferences in maneuver choice," in *Proc. 49th IEEE Conf. Decis. Control*, 2010, pp. 1189–1194.

[31] Z.-H. Mao, E. Feron, and K. Bilimoria, "Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 101–109, Jun. 2001.

[32] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Collision avoidance in air traffic management: A mixed-integer linear optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 47–57, Mar. 2011.

[33] R. A. Paielli and H. Erzberger, "Conflict probability for free flight," *J. Guid., Control, Dyn.*, vol. 20, no. 3, pp. 588–596, May 1997.

[34] L. Pallottino, E. M. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 3–11, Mar. 2002.

[35] E. Lalish, K. A. Morgansen, and T. Tsukamaki, "Decentralized reactive collision avoidance for multiple unicycle-type vehicles," in *Proc. Amer. Control Conf.*, 2008, pp. 5055–5061.

[36] M. Ishutkina, E. Feron, and K. Bilimoria, "Describing air traffic complexity using mathematical programming," in *Proc. AIAA 5th ATIO 16th Lighter-Than-Air Syst. Tech. Balloon Syst. Conf.*, Jun. 2012, pp. 1–9.

[37] K. Lee, E. Feron, and A. Pritchett, "Describing airspace complexity: Airspace response to disturbances," *J. Guid., Control, Dyn.*, vol. 32, no. 1, pp. 210–222, Jan. 2009.

[38] E. Frazzoli, Z. H. Mao, J. H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *J. Guid., Control, Dyn.*, vol. 24, no. 1, pp. 79–86, Jan. 2001.

[39] Z.-H. Mao, D. Dugail, and E. Feron, "Space partition for conflict resolution of intersecting flows of mobile agents," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 512–527, Sep. 2007.

[40] Z.-H. Mao, E. Feron, and K. Bilimoria, "Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 101–109, Jun. 2001.

[41] Z.-H. Mao and E. Feron, "Stability and performance of intersecting aircraft flows under sequential conflict resolution," in *Proc. Amer. Control Conf.*, 2001, pp. 722–729.

[42] Z.-H. Mao, D. Dugail, E. Feron, and K. Bilimoria, "Stability of intersecting aircraft flows using heading-change maneuvers for conflict avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 4, pp. 357–369, Dec. 2005.

[43] O. J. Boxma, O. Kella, and K. Kosiński, "Queue lengths and workloads in polling systems," *Oper. Res. Lett.*, vol. 39, no. 6, pp. 401–405, 2011.

[44] V. M. Vishnevskii and O. V. Semenova, "Mathematical methods to study the polling systems," *Autom. Remote Control*, vol. 67, no. 2, pp. 173–220, Feb. 2006.

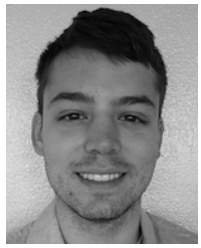
[45] H. Takagi, "Queueing analysis of polling models," *ACM Comput. Surv.*, vol. 20, no. 1, pp. 5–28, Mar. 1998.

[46] M. A. Boon, R. Van der Mei, and E. M. Winands, "Applications of polling systems," *Surv. Oper. Res. Manage. Sci.*, vol. 16, no. 2, pp. 67–82, 2011.

[47] H. Levy and M. Sidi, "Polling systems: Applications, modeling, and optimization," *IEEE Trans. Commun.*, vol. 38, no. 10, pp. 1750–1760, Oct. 1990.

[48] Z. Liu, P. Nain, and D. Towsley, "On optimal polling policies," *Queueing Syst.*, vol. 11, no. 1/2, pp. 59–83, 1992.

[49] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 1417–1423.
[50] J. Gregoire and E. Frazzoli, "Hybrid centralized/distributed autonomous intersection control: Using a job scheduler as a planner and inheriting its efficiency guarantees," in *Proc. IEEE 55th Conf. Decis. Control.*, 2016, pp. 2549–2554.
[51] E. Altman and D. Kofman, "Bounds for performance measures of token rings," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 292–299, Apr. 1996.
[52] E. Altman, S. Foss, E. Riehl, and S. Stidham, Jr., "Performance bounds and pathwise stability for generalized vacation and polling systems," Dept. Oper. Res., Univ. North Carolina Chapel Hill, Tech. Rep. UNC/OR TR93-8, 1993.
[53] R. L. Cruz, "A calculus for network delay. I. Network elements in isolation," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.
[54] D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, 2nd ed. Hoboken, NJ, USA: Wiley, 1995.
[55] J. Teichmann, F. Ballani, and K. G. van den Boogaart, "Generalizations of Matern's hard-core point processes," *Spatial Statist.*, vol. 3, pp. 33–53, Sep. 2012.
[56] F. Baccelli and P. Bermolen, "Extremal versus additive Matérn point processes," *Queueing Syst.*, vol. 71, no. 1/2, pp. 179–197, Mar. 2012.
[57] D. Stoyan and H. Stoyan, "On one of matern's hard-core point process models," *Math. Nachr.*, vol. 122, pp. 205–214, Jan. 1985.
[58] D. L. Gerlough and M. J. Huber, "Traffic flow theory," Transp. Res. Board, Tech. Rep. No. HS-006 783, 1976.
[59] M. E. Fouladvand, Z. Sadjadi, and M. R. Shaebani, "Characteristics of vehicular traffic flow at a roundabout," *Phys. Rev. E*, vol. 70, no. 4, 2004, Art. no. 046132.
[60] M. E. Fouladvand, Z. Sadjadi, and M. R. Shaebani, "Optimized traffic flow at a single intersection: Traffic responsive signalization," *J. Phys. A, Math. Gen.*, vol. 37, no. 3, 2004, Art. no. 561.
[61] M.-B. Hu, R. Jiang, R. Wang, and Q.-S. Wu, "Urban traffic simulated from the dual representation: Flow, crisis and congestion," *Phys. Lett. A*, vol. 373, no. 23/24, pp. 2007–2011, 2009.
[62] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com
[63] S. C. Borst, "Polling systems with multiple coupled servers," *Queueing Syst.*, vol. 20, no. 3, pp. 369–393, 1995.
[64] P. Beekhuizen, T. Denteneer, and J. Resing, "End-to-end delays in polling tree networks," in *Proc. 3rd Int. Conf. Perform. Eval. Methodologies Tools*, 2008, p. 42.
[65] L. van den Bos and M. Boon, "Networks of polling systems," Technische Universiteit Eindhoven, Eindhoven, The Netherlands, Bachelor's thesis, 2013.
[66] D. Miculescu and S. Karaman, "Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals," Jul. 2016, arXiv:1607.07896.



David Miculescu received the B.S. degree in aerospace engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2013. He received the S.M. degree in aerospace engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2015, where he is currently working toward the Ph.D. degree.

His research interests include the general areas of autonomous networks and control systems.

Mr. Miculescu is a National Science Foundation Graduate Research Fellow.

Sertac Karaman received the S.M. degree in mechanical engineering and the Ph.D. degree in electrical engineering and computer science both from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009 and 2012, respectively.

He is currently an Associate Professor of Aeronautics and Astronautics at the Massachusetts Institute of Technology (since Fall 2012). His research interests include the broad areas of robotics and control theory. In particular, he studies the applications of probability theory, stochastic processes, stochastic geometry, formal methods, and optimization for the design and analysis of high-performance cyber-physical systems.

Dr. Karaman is the recipient of the IEEE Robotics and Automation Society Early Career Award in 2017, an Office of Naval Research Young Investigator Award in 2017, an Army Research Office Young Investigator Award in 2015, and National Science Foundation Faculty Career Development Award in 2014.