

Safety Verification and Control for Collision Avoidance at Road Intersections

by

Heejin Ahn

B.S., Seoul National University (2012)

S.M., Massachusetts Institute of Technology (2014)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Signature redacted


Author

Department of Mechanical Engineering

February 16, 2018

Signature redacted

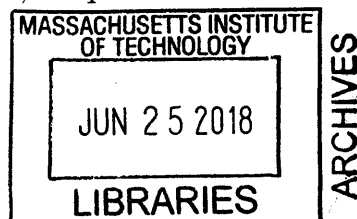
Certified by


Domitilla Del Vecchio
Professor, Department of Mechanical Engineering
Thesis Supervisor

Signature redacted

Accepted by


Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses



Safety Verification and Control for Collision Avoidance at Road Intersections

by

Heejin Ahn

Submitted to the Department of Mechanical Engineering
on February 16, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Car crashes cause a large number of fatalities and injuries, with about 33,000 people killed and 2.3 million injured in the United States every year. To prevent car crashes, the government and automotive companies have taken initiatives to develop and deploy communications among vehicles and between vehicles and infrastructure. By using such communications, we design centralized coordinators at road intersections, called *supervisors*, that monitor the dynamical state of vehicles and the current input of drivers and override them if necessary to prevent a collision. The primary technical problem in the design of such systems is to determine if the current drivers' input will cause an unavoidable future collision, in which case the supervisor must override the drivers at the current time to prevent the collision. This problem is called *safety verification problem* which is known to be computationally intractable for general dynamical systems. Our approach to solving the safety verification problem is to translate it to a computationally more tractable scheduling problem. When modeling an intersection as a single conflict area inside which the paths of vehicles intersect, we exactly solve the scheduling problem with algorithms that can handle a small number of vehicles in real-time. For a larger number of vehicles or with more complex intersection models, we approximately solve it within quantified approximation bounds by using mixed integer linear programming (MILP) formulations that, despite the combinatorial complexity, can be solved in real-time by available software such as CPLEX. Based on the solutions to the safety verification problem, we design a supervisor and prove that it ensures safety and is nonblocking, another major challenge of verification-based algorithms. We validate the supervisor using computer simulations and experiments.

Thesis Supervisor: Domitilla Del Vecchio

Title: Professor, Department of Mechanical Engineering

Acknowledgments

I am truly indebted to a lot of people for their support and encouragement during my Ph.D.

First, I would like to thank my thesis advisor, Prof. Domitilla Del Vecchio. She was a charismatic leader and a patient teacher, and I especially appreciate her guidance on my research direction and helpful comments on my papers and presentations. I am also thankful to Dr. Karl Iagnemma and Prof. Sertac Karaman in my thesis committee for sharing their insights in the field of autonomous driving and showing interests in my Ph.D. thesis. This thesis would not be possible without the help of Prof. Alessandro Colombo at Politecnico di Milano. Working with him at Milan, Italy, on one chapter of this thesis was one of the most valuable experiences during my Ph.D., thanks to him and the students at Politecnico di Milano.

Second, I would like to thank my fellow graduate students and postdocs in the DDV group for sharing my ups and downs during my graduate study. Also, I would like to express my gratitude to my Korean friends at MIT, including my KGSAME and GSTS friends. My special thank goes to Sunyoung Park, who, now at Harvard, has been a friend whom I can always count on since 2005.

Last but not least, I would like to thank my family in Korea for their constant love and encouragement. I am most grateful to my husband, Minkyun Noh, who was a boyfriend when I wrote my master thesis at MIT a few years ago and since then, has been a great motivator and partner. Thanks to God for always being with me wherever I am.

Contents

1	Introduction	19
1.1	Motivation	19
1.2	Statement of Contributions	22
2	Problem Setup and Statement	29
2.1	Intersection Models	29
2.1.1	Single Conflict Area ($m = 1$)	30
2.1.2	Multiple Conflict Areas ($1 < m < \infty$)	30
2.1.3	Multiple Conflict Areas with Merging/Splitting Paths ($m = \infty$)	31
2.2	Vehicle Dynamics	33
2.3	Supervisor Design and Safety Verification	34
3	Preliminary Results	37
3.1	Scheduling	37
3.1.1	Single-machine Scheduling	38
3.1.2	Jobshop Scheduling	39
3.2	Equivalence	40
3.3	Example	40
4	Supervisors at Single Conflict Area	43
4.1	Uncertainty Sources	44
4.2	Supervisor Design	46

4.2.1	Estimation	46
4.2.2	Prediction	47
4.2.3	Verification	48
4.3	Least Restrictive Supervisor	49
4.3.1	Inserted Idle-Time Scheduling Problem	49
4.3.2	Least Restrictive Supervisor Algorithm	53
4.4	Approximate Supervisor	55
4.4.1	Approximate Verification	55
4.4.2	Approximate Supervisor Algorithm	61
4.5	Validation of the Supervisors	63
4.5.1	Computer Simulations	63
4.5.2	Experiments	65
5	Supervisor at Multiple Conflict Areas	71
5.1	Jobshop Scheduling	73
5.2	Approximate Verification	79
5.2.1	Lower Bound Problem	79
5.2.2	Upper Bound Problem	82
5.2.3	Approximation Bounds	86
5.3	Supervisor Algorithm	89
5.4	Validation of the Supervisor: Simulations	92
6	Supervisor at Multiple Conflict Areas and Merging/Splitting Paths	97
6.1	Comparison of Discretization Schemes	100
6.1.1	Time Discretization	100
6.1.2	Space Discretization	102
6.2	Verification in a Hierarchical Structure	103
6.2.1	Verification on the Abstract System	103
6.2.2	Verification on the Concrete System	109
6.3	Supervisor Algorithm	116
6.4	Validation of the Supervisor: Simulations	118

6.4.1	Implementation of the Supervisor Algorithm	118
6.4.2	Performance Comparisons	121
7	Conclusion	125
7.1	Discussion	126
7.2	Future Directions	128
A	Proofs in Chapter 4	131
B	Proofs in Chapter 5	139

List of Figures

1-1	As opposed to the automated intersection management framework in (a), we adopt the semiautonomous framework in (b) where the system should decide when to change modes.	20
1-2	If a state is inside the backward reachable set of the bad set, all possible trajectories starting from the state enters the bad set.	21
2-1	Single conflict area (dotted oval) and multiple conflict areas (red shaded circles). If modeling the intersection as a single conflict area, we can simplify the scenario in (a) to the scenario in (b). By modeling the intersection as a set of multiple conflict areas, we can design a less restrictive supervisor.	30
2-2	Realistic complex intersection scenario. We design a supervisor that overrides these vehicles when necessary to prevent side or rear-end collisions. We assume that the paths of vehicles are known in advance; gray vehicles will go straight while white vehicles will turn.	32
2-3	Supervisor returns the desired input \mathbf{u}_d if the input will not cause a future collision, and otherwise returns a safe input \mathbf{u}_s	35
3-1	Example of the graph representation.	39

4-1	Road intersection (left) and its simplified model (right). An intersection is modeled as a conflict area (the shaded area), and its location is denoted by an open interval (α_i, β_i) along the path of car i . The supervisor communicates with and controls only cars 1 and 2 to avoid a collision among all three cars.	43
4-2	Structure of the supervisor. The verification algorithm predicts whether the supervisor should override controlled vehicles with a safe input \mathbf{u}_s to prevent collisions.	46
4-3	In simulation, the approximate supervisor intervenes no earlier than the least restrictive supervisor. The solid black and dotted red lines represent the actual position of the controlled and uncontrolled vehicles, respectively, with gray surrounding representing the state estimate. The blue bands on the bottom appear when the supervisor overrides the controlled vehicles to prevent a collision at the intersection at $(0, 65)$	64
4-4	The blue bands at the bottom show that our supervisor in (b) starts an override later than the supervisor in (a). The same coloring is used as in Fig. 4-3	65
4-5	Experimental setup. The supervisor modifies the inputs of cars 1 and 2 only when necessary to avoid collisions at the intersection (shaded circle) in the presence of an uncontrolled vehicle (car 3).	65
4-6	Experimental results of the supervisor. The solid black and dotted red lines represent the position measurement of the controlled and uncontrolled cars, respectively. The same coloring is used as in Figure 4-3.	67
4-7	Position trajectories of cars 1–3. The red region represents a set of positions where cars collide. The trajectories (semitransparent black lines) are in blue when the supervisor intervenes. The 2D projections confirm that no trajectory enters the red region.	69

5-1	An intersection is modeled as a set of conflict areas near which two longitudinal predefined paths intersect. The locations of conflict areas 1 and 3 along the path of vehicle 1 are denoted by $(\alpha_{1,1}, \beta_{1,1})$ and $(\alpha_{3,1}, \beta_{3,1})$, respectively.	71
5-2	General intersection. The safety verification problem in this scenario, which involves 20 vehicles and 48 conflict areas, without considering rear-end collision avoidance on merging paths, can be approximately solved within quantified bounds. This intersection is obtained from [41] to encompass 20 top crash locations in Massachusetts, USA.	72
5-3	(a) All operations of the scenario in Figure 5-1. (b) Operations when $\beta_{1,1} \leq x_1(0) < \beta_{3,1}, x_2(0) < \beta_{2,2}$, and $\beta_{3,3} \leq x_3(0) < \beta_{2,3}$. The black solid and red dotted arrows are conjunctive and disjunctive arcs, respectively. See Example 5.1 for more details.	75
5-4	Illustration of Definitions 5.1 and 5.2, where the x -axis represents time and the y -axis represents the position. Suppose $(i', j) \in \mathcal{F}$ and $(i', j) \rightarrow (i, j) \in \mathcal{C}$. We can compute $\bar{P}_{i',j}, \bar{T}_{i,j}, \bar{P}_{i,j}$ by considering the maximum input inside the intersection.	83
5-5	Shrunk and Inflated conflict areas for $(i', j) \in \mathcal{F}$ and $(i', i) \in \mathcal{C}_j$. Figures (a)-(d) illustrate (5.13)-(5.16), respectively. By definition, $(\hat{\alpha}_{i',j}, \hat{\beta}_{i',j}) \subseteq (\alpha_{i',j}, \beta_{i',j}) \subseteq (\check{\alpha}_{i',j}, \check{\beta}_{i',j})$ and $(\hat{\alpha}_{i,j}, \hat{\beta}_{i,j}) \subseteq (\alpha_{i,j}, \beta_{i,j}) \subseteq (\check{\alpha}_{i,j}, \check{\beta}_{i,j})$	87
5-6	The supervisor overrides the vehicles if Problem 5.3 returns $s_U^* > 0$, given the desired state $(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k)$	90
5-7	Simulation results without the supervisor (Algorithm 5-7) for the scenario in Figure 5-1. Cases I, II, and III denote the same cases in Table 5.1.	92

5-8	Simulation results with the supervisor for the scenario in Figure 5-1. The black line represents the trajectory and is the same on each figure. The line turns to the dotted line when the supervisor intervenes to prevent a predicted collision. The solid is (a) the inflated bad set, (b) the bad set, and (c) the shrunken bad set. The supervisor manages the system to avoid entering the bad set.	95
5-9	Trajectory of vehicle 1 in the scenario of Figure 5-2, which involves 20 vehicles, 48 conflict areas, and 120 operations. The blue boxes represent the times at which the supervisor overrides the vehicles. The red dotted lines are the trajectories of the other vehicles that share the same conflict area. This graph shows that each conflict area is used by only one vehicle at a time.	96
6-1	Scenario with four vehicles. The paths of vehicles intersect inside four side conflict areas and partly overlap.	97
6-2	Hierarchical structure for the safety verification problem. The verification on an abstract system is used for the verification on an concrete system.	98
6-3	Discretized path $\{\{\xi_j[k-1], \xi_j[k]\}_{k=1}^{13}$ of vehicle j and sequence of time intervals Δt_j , based on which a trajectory of the abstraction (red line) is defined.	109
6-4	With $\Delta v_{\max} = 0$, the smoothing function that satisfies (6.15) with equality is represented by the black solid line, and a linear smoothing function that satisfies (6.15) (and thus Condition 6.2) by the red dotted line.	114
6-5	With λ, η , and h satisfying Conditions 6.1 and 6.2, a trajectory of the concrete system tracks an abstraction trajectory within $e_{\max} = 1$, and the magnitude of s decreases over a segment. The vertical dotted lines indicate time intervals taken to travel each segment.	119

6-6	Trajectory of vehicle 16 through two rear-end conflict areas ($O_{16,7}$ and $O_{16,9} = O_{16,10}$) and five side conflict areas, as depicted in Figure 2-2. The dotted lines represent the trajectories of other vehicles that share the same rear-end or side conflict area. Vehicles maintain a safe distance of 4 on the rear-end conflict areas, and are not simultaneously present inside each side conflict area.	120
6-7	Input signal applied to vehicle 16. The supervisor overwrites the desired input (black line) when a future collision is detected. The dotted red line indicates the sliding mode control input applied by the supervisor.	120
6-8	Computation time to solve Problem 6.1 as the number of vehicles in the scenario of Figure 2-2 increases.	121
6-9	As varying \dot{x}_1 and \dot{x}_4 from $\dot{x}_{\min} = 1$ to $\dot{x}_{\max} = 15$, we plot a white dot if there is a feasible solution to Problem 6.1 given a state $\mathbf{x}(0) = (-15, -6, -15, -15)$ and $\dot{\mathbf{x}}(0) = (\dot{x}_1, 15, 15, \dot{x}_4)$, and a red dot otherwise. We use $e_{\max} = 1$ in (a) and 0.25 in (b). The smaller number of red dots in (b) indicates that our problem on the finer grid is less restrictive while requiring more computation time.	123
6-10	Results of time discretization-based verification, with different time steps. Our approach in Figure 6-9 is not more restrictive than the time discretization-based verification while computationally favorable.	124
7-1	Paths of vehicles are undetermined before they enter the intersection.	127
7-2	Sequence of discrete modes in city driving scenarios. LF_i is the lane following mode in lane i and S_i is the stop mode in lane i	129

List of Algorithms

4-1	Solution to Problem 4.2	52
4-2	Implementation of the least restrictive supervisor at time $k\tau$	54
4-3	Modified version of the result of [27]	56
4-4	Solution to Problem 4.3	58
4-5	Approximate Solution of Problem 4.1	59
4-6	Implementation of an approximate supervisor	62
5-7	Supervisory control algorithm at $t = k\tau$	90
6-8	Implementation of the supervisor at time step k	116

Chapter 1

Introduction

In this chapter, we introduce motivations for the designs of driver assistance systems that ensure safety of vehicles at road intersections, and state the contributions of this thesis.

1.1 Motivation

According to the National Highway Traffic Safety Administration (NHTSA), 33,561 people died, and 2,362,000 people were injured in the United States in 2012 because of car crashes [61]. To reduce the number of car crashes, the U.S. Department of Transportation (USDOT) and automotive companies have started initiatives to deploy vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication technology to allow vehicles to exchange information with each other [60]. This communication technology stimulated research on networked vehicle systems to maintain safety, mostly on highways, since the 90's with the California PATH project [9,10]. More recently, the communication technology has been utilized to prevent collisions among multiple vehicles at road intersections [24,55]. This thesis proposes the design of centralized coordinators that ensure collision-free intersections by exploiting the communication technology.

A common framework to implement centralized coordinators at road intersections is an *automated intersection management*, where the coordinators take full control of

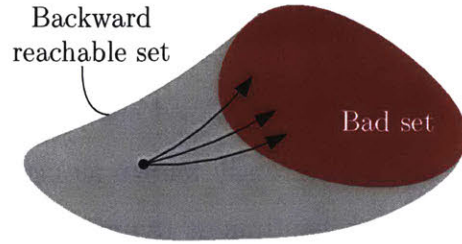


Figure 1-2: If a state is inside the backward reachable set of the bad set, all possible trajectories starting from the state enters the bad set.

to calculate the backward reachable set of the bad set, which is defined as the set of starting states from which state trajectories enter the collision set independent of the control input [39,42,57,59]. The backward reachable set of the bad set is illustrated in Figure 1-2. By definition, if a given initial state is inside this set, there is no control input that prevents the state trajectory from entering the bad set eventually. To ensure safety, the coordinators in the semiautonomous framework should override vehicles before the state enters the backward reachable set of the bad set.

However, reachability analysis of dynamical systems with large state spaces is usually challenging due to the complexity of computing reachable sets. This motivated the development of several approximation approaches. One approximation approach is to consider a simplified dynamical model to compute reachable sets instead of using the original complex dynamical model, as studied in [8,29,32]. Another approach is to approximately represent the original reachable set by employing various geometric objects, including polyhedra [12], ellipsoids [26], or parallelotopes [23]. The reachability analysis becomes relatively simple if system dynamics are monotone, which means that state trajectories preserve a partial ordering on states and inputs [43]. This is because, for such systems, the boundary of a reachable set can be computed by considering only maximum and minimum states and inputs [22]. Indeed, an exact method for the reachability analysis is presented in [28,31] for monotone systems.

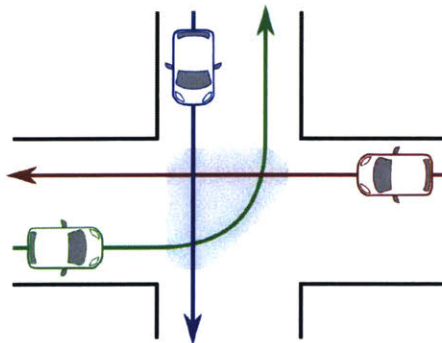
Our approach to designing a centralized coordinator in the semiautonomous framework relies on the monotonicity of dynamics and the approximation of dynamics. The detailed description of our approach and the contributions of the thesis will be discussed in the following section.

1.2 Statement of Contributions

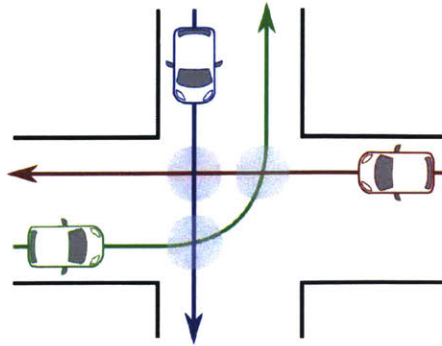
Systems consisting of vehicles and a centralized coordinator in the semiautonomous framework are *hybrid systems*, also recently known as *cyber-physical systems* (CPS), because the systems involve both continuous dynamical states and discrete modes. The objective of this thesis is to design a centralized coordinator that returns both a discrete input (decision for mode transition) and a continuous input (control input to the vehicle dynamics) to ensure safety at road intersections. More precisely, as illustrated in Figure 1-1, the coordinator is in the SILENT mode unless the current input of drivers will cause any future collision. In the OVERRIDE mode, the coordinator takes control of the drivers and applies a safe continuous input. Such a coordinator is referred to as a *supervisor* [52].

The supervisor determines the timing of mode transition by solving the safety verification problem, as discussed in the previous section. We exploit the monotonicity of the longitudinal dynamics of vehicles to translate the safety verification problem into a computationally more tractable scheduling problem. The safety verification problem is equivalent to a single-machine scheduling problem if an intersection is modeled as a single area inside which all the longitudinal paths of vehicles intersect, and equivalent to a jobshop scheduling problem if an intersection is modeled as multiple areas inside which some pairs of the paths intersect. That is, the safety verification problem is rewritten as a different scheduling problem depending on an intersection model. In the thesis, we separately consider three intersection models of different complexity levels:

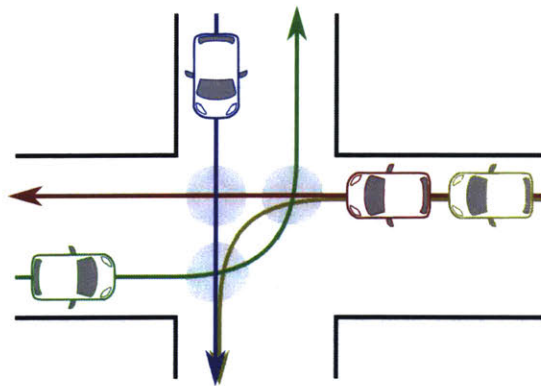
Model 1: The paths of vehicles intersect inside a single conflict area.



Model 2: The paths of vehicles intersect inside multiple conflict areas.



Model 3: The paths of vehicles intersect inside multiple conflict areas and partly overlap.



With the simple model (model 1), a collision occurs if more than one vehicle stays inside the single conflict area at the same time. Supervisor algorithms that prevent such a collision were previously studied in [16,17]. As opposed to the previous works, this thesis considers uncertainty sources and designs a *robust supervisor* to allow its practical implementation. More precisely, in this problem setting, the contributions of the thesis are as follows.

- We present the design of a provably safe, least restrictive supervisor that handles the simultaneous presence of measurement errors, unmodeled dynamics, and uncontrolled vehicles, which are not equipped with communication modules and thus not controlled by the supervisor.

- Because the algorithm of the least restrictive supervisor has combinatorial complexity with the number of vehicles, we also present a novel strategy to design an approximate supervisor that can handle a larger number of vehicles in polynomially bounded time.
- We quantify an approximation error bound of the approximate supervisor.
- We validate the least restrictive supervisor through experiments using three in-scale cars and the approximate supervisor through computer simulations.

The works in [1,2,14] presented methods to separately handle the errors in measurement and dynamical model, and the presence of uncontrolled vehicles. By merging the methods, this thesis presents a method that handle the simultaneous presence of uncertainties. Also, the approximate supervisor designed in the thesis is less restrictive than the previously proposed approximate supervisors [1,2,14]. That is, the approximate supervisor of this thesis overrides drivers less often than the previously proposed ones.

With a more complex model (model 2), a collision occurs if two vehicles are simultaneously inside the same conflict area. Since an equivalent jobshop scheduling problem is computationally intractable due to its nonconvexity and nonlinearity, the thesis instead presents approximate solutions to the problem within quantified approximation bounds. The contributions of the thesis in this setting are as follows.

- We prove that the safety verification problem is equivalent to a jobshop scheduling problem for general longitudinal dynamics (second-order and nonlinear).
- Because the jobshop scheduling problem is a mixed integer nonlinear programming (MINLP) problem due to the second-order dynamics of vehicles, we approximately solve the jobshop scheduling problem by formulating two mixed integer linear programming (MILP) problems based on first-order linear dynamics and second-order nonlinear dynamics on a restricted input space, respectively.
- We prove that the two MILP problems provide over- and under-approximations

of the exact solution to the jobshop scheduling problem, and quantify the approximation error bounds.

- We design a supervisor algorithm based on an approximate solution that ensures safety, and validate it through computer simulations.

A similar problem where robots follow predefined paths is considered in [48,49], but the approach of these papers is not designed for safety verification and thus is restricted to zero initial speed with double integrator dynamics. Our scheduling approach can deal with general vehicle dynamics and verify safety for any given starting state.

In general and realistic intersection models (model 3), the paths of vehicles define *side conflict areas* as the areas inside which the paths intersect, and *rear-end conflict areas* as the regions where two paths overlap. Here, a side collision occurs if two vehicles meet inside a side conflict area, and a rear-end collision occurs if the distance between two vehicles in a rear-end conflict area is less than a minimum safe distance. In order to construct a supervisor algorithm that prevents both rear-end and side collisions, we adopt a *hierarchical structure*. On the top layer of the structure, we formulate a scheduling problem based on a coarse dynamical model, referred to as an *abstraction* or *abstract system*. On the low layer, the goal is to find a solution to the safety verification problem based on actual vehicle dynamics, referred to as the *concrete system*. Here, the main contribution of the thesis is that *we provide a map between the solutions to a scheduling problem formulated in terms of the abstract system and to the safety verification problem formulated in terms of the concrete system*. To achieve this, the thesis presents the following.

- We formulate a novel scheduling problem based on an abstract system of the actual longitudinal vehicle dynamics and on discretized paths of vehicles. The solution to the scheduling problem gives a trajectory of the abstract system that avoids the bad set inflated by a given value e_{\max} .
- By appropriately designing the interface between the abstract and concrete systems using sliding mode control, we construct a trajectory of the concrete

system that tracks the trajectory of the abstract system within a given error bound e_{\max} . By construction, the trajectory of the concrete system never enters the bad set.

- We design a supervisor algorithm based on the solution to the scheduling problem (thus the solution to the safety verification problem), and validate its computationally efficient performance through computer simulations.
- We compare our approach with another approach based on time discretization to show through computer simulations the computational benefit of our approach while maintaining similar restrictiveness.

Uniform time discretization was used to design a centralized coordinator that maintains safety at intersections in the automated framework [34] and in the semiautonomous framework [6,7]. While formally correct, these solutions are computationally demanding, because of the structure of the ensuing optimization problems; we will discuss this in detail in Chapter 6. Thus, our approach is based on discretizing space rather than time, which is a better approach from the computational point of view. Similar approaches based on discretization and abstraction were studied in [15,20,21]. Using time and input space discretization, the work in [15] performed the abstraction by exploiting the differentially flat systems, and the works in [20,21] constructed a discrete event system (DES) abstraction and applied supervisory control theory of DES. These approaches, however, are applicable only for a single conflict area and require higher computation costs than our approach.

The rest of the thesis is organized as follows. In Chapter 2, we define the bad set (i.e., the set of points corresponding to collisions) for each intersection model, explain the longitudinal vehicle dynamics, and state the design specifications for the supervisor based on safety verification. In Chapter 3, we present the background of scheduling problems and the formal definition of equivalence between two decision problems, and provide an example of a single-machine scheduling problem that is equivalent to the safety verification problem at a single conflict area. In Chapters 4-6, we design supervisor algorithms in the three different intersection models, respec-

tively. In each chapter, we formulate a scheduling problem, prove that the resulting supervisors ensure safety at intersections and are nonblocking, and validate the supervisors via computer simulations and/or experiments. We conclude the thesis in Chapter 7 by summarizing the thesis and suggesting future research directions.

Chapter 2

Problem Setup and Statement

Throughout the thesis, we assume that the paths of vehicles are known before they enter an intersection, for example, via road regulation that allows a specific turn on each road or via estimation algorithms (e.g., [40]) that enable the prediction of future paths. Depending on the number of points at which the paths intersect, we characterize three intersection models of different complexity levels. In this section, we also describe the longitudinal dynamics of vehicles and their property, and state the safety verification problem and the design specifications of the supervisor.

2.1 Intersection Models

Consider that n vehicles are approaching an intersection. The position of vehicles' centers on their own longitudinal paths is denoted by $\mathbf{x} = (x_1, x_2, \dots, x_n) \in X_1 \times X_2 \times \dots \times X_n = \mathbf{X} \subset \mathbb{R}^n$. A *conflict area* refers to an area around a location at which the longitudinal paths of vehicles intersect, and its size is determined by the lengths of vehicles. In this section, we introduce three intersection models in terms of the number of conflict areas m .

2.1.1 Single Conflict Area ($m = 1$)

In the simplest scenario, an intersection is modeled as a single conflict area in which all the paths of vehicles intersect, as illustrated by a dotted oval in Figure 2-1(a). We assume in this scenario that there is only one vehicle per road to concentrate only on the avoidance of side collisions. We can extend the result to incorporate the avoidance of rear-end collisions by using a method given in [17]. This scenario can be equivalently illustrated by Figure 2-1(b).

Let $(\alpha_i, \beta_i) \subset X_i \subset \mathbb{R}$ be the location of the conflict area on the path of vehicle i . Then, the set of points corresponding to side collisions is

$$B := \{\mathbf{x} \in \mathbf{X} : x_i \in (\alpha_i, \beta_i), x_j \in (\alpha_j, \beta_j) \text{ for some } i \neq j\}, \quad (2.1)$$

and called the *bad set*.

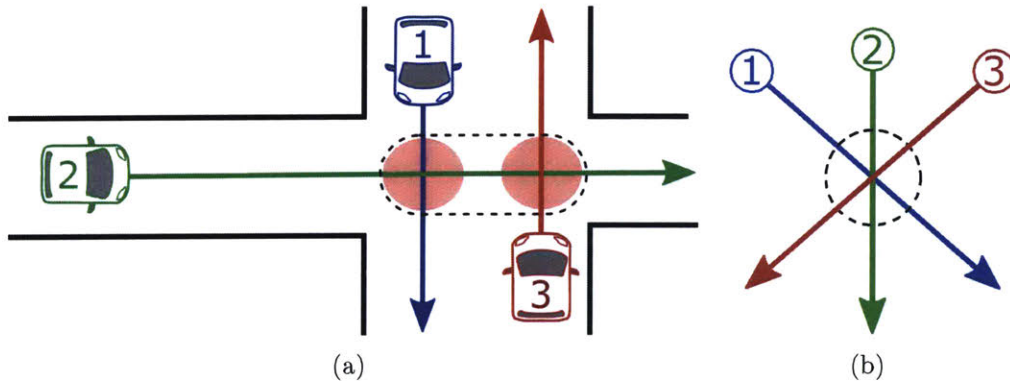


Figure 2-1: Single conflict area (dotted oval) and multiple conflict areas (red shaded circles). If modeling the intersection as a single conflict area, we can simplify the scenario in (a) to the scenario in (b). By modeling the intersection as a set of multiple conflict areas, we can design a less restrictive supervisor.

2.1.2 Multiple Conflict Areas ($1 < m < \infty$)

In the single conflict area scenario depicted in Figure 2-1(a), the supervisor prevents cars 1 and 3 from simultaneously staying inside the conflict area (dotted oval) although their collision is geometrically infeasible. To design a less restrictive supervisor, we model the intersection as a set of distinct conflict areas (red shaded circles). We

still assume that there is only one vehicle per road so as to focus on side collision avoidance only.

Conflict area i on the path of vehicle j is located at $(\alpha_{i,j}, \beta_{i,j}) \subset X_j \subset \mathbb{R}$. Let $(j, j') \in \mathcal{D}_i$ denote that the paths of vehicles j and j' intersect inside conflict area i . Then, the bad set, which is the set of points corresponding to side collisions, is

$$B := \{\mathbf{x} \in \mathbf{X} : x_j \in (\alpha_{i,j}, \beta_{i,j}), x_{j'} \in (\alpha_{i,j'}, \beta_{i,j'}) \text{ for some } (j, j') \in \mathcal{D}_i\}. \quad (2.2)$$

For example, in Figure 2-1(a), say that the left circle is conflict area 1 and the right circle is conflict area 2. We have $(1, 2) \in \mathcal{D}_1$ and $(2, 3) \in \mathcal{D}_2$, and the bad set is

$$B = \{(x_1, x_2, x_3) \in \mathbf{X} : (x_1 \in (\alpha_{1,1}, \beta_{1,1}) \text{ and } x_2 \in (\alpha_{1,2}, \beta_{1,2})) \\ \text{or } (x_2 \in (\alpha_{2,2}, \beta_{2,2}) \text{ and } x_3 \in (\alpha_{2,3}, \beta_{2,3}))\}.$$

2.1.3 Multiple Conflict Areas with Merging/Splitting Paths ($m = \infty$)

As illustrated in Figure 2-2, a realistic intersection model considers rear-end conflict areas as well as multiple side conflict areas. If the paths of two vehicles overlap, then the vehicles must maintain a minimum safe distance d in the rear-end conflict area. In this model, we relax the assumption of one vehicle per road and consider rear-end collision avoidance in conjunction with side collision avoidance.

When the paths of vehicles j and j' overlap, we say $(j, j') \in \mathcal{O}$. A rear-end conflict area is defined as a closed interval with a nonempty interior where the two paths overlap. The location of a rear-end conflict area is denoted by $O_{j,j'} \subset X_j \subset \mathbb{R}$ on the path of vehicle j . Let $o_{j,j'} := \min O_{j,j'}$. The sets of points corresponding to side collisions and to rear-end collisions are

$$B_{\text{side}} := \{\mathbf{x} \in \mathbf{X} : x_j \in (\alpha_{i,j}, \beta_{i,j}), x_{j'} \in (\alpha_{i,j'}, \beta_{i,j'}) \text{ for some } (j, j') \in \mathcal{D}_i\},$$

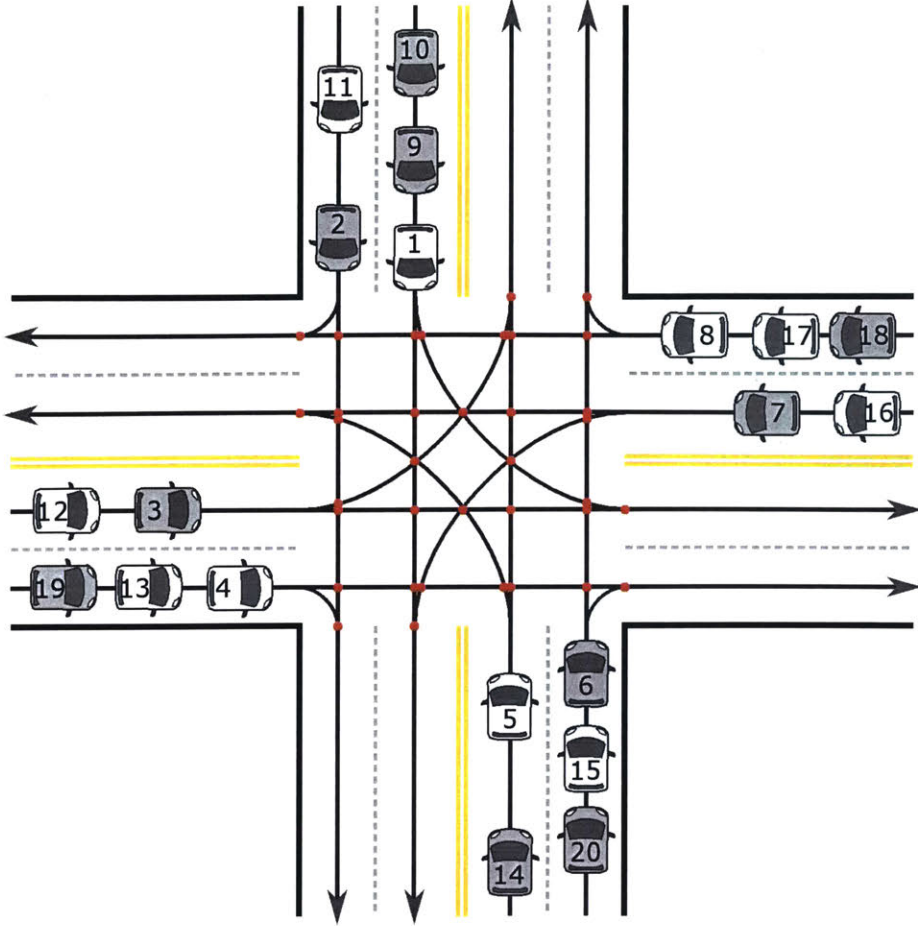


Figure 2-2: Realistic complex intersection scenario. We design a supervisor that overrides these vehicles when necessary to prevent side or rear-end collisions. We assume that the paths of vehicles are known in advance; gray vehicles will go straight while white vehicles will turn.

and

$$B_{\text{rear-end}} := \{ \mathbf{x} \in \mathbf{X} : \text{for some } (j, j') \in \mathcal{O} \text{ if } x_j \in O_{j,j'} \text{ and } x_{j'} \in O_{j',j}, \\ |(x_j - o_{j,j'}) - (x_{j'} - o_{j',j})| < d \},$$

respectively. The union of these sets,

$$B = B_{\text{side}} \cup B_{\text{rear-end}}, \tag{2.3}$$

is the bad set.

2.2 Vehicle Dynamics

The dynamical state of vehicle j is (x_j, \dot{x}_j) where $x_j \in X_j \subset \mathbb{R}$ is the position and $\dot{x}_j \in \dot{X}_j := [\dot{x}_{j,\min}, \dot{x}_{j,\max}] \subset \mathbb{R}$ is the speed along its longitudinal path. The longitudinal dynamics of vehicle j are

$$\ddot{x}_j = f_j(x_j, \dot{x}_j, u_j), \quad (2.4)$$

where $u_j \in U_j := [u_{j,\min}, u_{j,\max}] \subset \mathbb{R}$ is the control input, which is a throttle or brake input. We assume that the differential equation (2.4) has a unique solution, and the solution depends continuously on the input. We consider scenarios where the radii of the paths of vehicles are sufficiently greater than the lengths of the vehicles, and the speeds of vehicles are low. In such scenarios, the coupling between the lateral and longitudinal dynamics can be ignored [31], which allows us to focus only on the control of longitudinal dynamics. For example, a standard longitudinal dynamical model is

$$\ddot{x}_j = c_1 u_j - c_2 (\dot{x}_j)^2 + c_3 \quad (2.5)$$

with parameters $c_1 > 0, c_2 > 0$ and c_3 . The nonlinear term appears due to aerodynamic drag force, and the constant term appears due to rolling resistance.

The whole system dynamics can be obtained by combining the individual dynamics (2.4) and written as

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}), \quad (2.6)$$

where $\mathbf{x} = (x_1, \dots, x_n) \in \mathbf{X} \subset \mathbb{R}^n$ and similarly, $\dot{\mathbf{x}} \in \dot{\mathbf{X}} = [\dot{\mathbf{x}}_{\min}, \dot{\mathbf{x}}_{\max}] \subset \mathbb{R}^n$, $\mathbf{u} \in \mathbf{U} = [\mathbf{u}_{\min}, \mathbf{u}_{\max}] \subset \mathbb{R}^n$.

The input $u_j(t)$ is a value of the input signal u_j evaluated at a time instance t , where an input signal $u_j : \mathbb{R}_+ \rightarrow U_j$ is a piecewise continuous function with a countable number of discontinuities. We assume that the space of input signals \mathcal{U}_j is connected. The function $x_j(t, u_j, x_j(0), \dot{x}_j(0))$ denotes the position reached at time t starting from $(x_j(0), \dot{x}_j(0))$ using an input signal $u_j \in \mathcal{U}_j$. For simplicity, we

write $x_j(t, u_j)$ if the initial condition is $(x_j(0), \dot{x}_j(0))$, and otherwise specify an initial condition. We also write the aggregate position $\mathbf{x}(t, \mathbf{u})$ with the aggregate input signal $\mathbf{u} \in \mathcal{U}_1 \times \cdots \times \mathcal{U}_n = \mathcal{U}$, implicitly assuming the initial condition $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$. Similarly, we use $\dot{\mathbf{x}}(t, \mathbf{u})$ to denote the speed at time t evolving with an input signal \mathbf{u} .

We now introduce the definition of monotonicity [11]. We say that an input signal u_j is no greater than another input signal u'_j , which is denoted by $u_j \leq u'_j$, if $u_j(t) \leq u'_j(t)$ for all $t \geq 0$.

Definition 2.1. The system (2.4) is monotone if $u_j \leq u'_j$, $x_j(0) \leq x'_j(0)$, $\dot{x}_j(0) \leq \dot{x}'_j(0)$, and $t \leq t'$,

$$x_j(t, u_j(\cdot), x_j(0), \dot{x}_j(0)) \leq x_j(t', u'_j(\cdot), x'_j(0), \dot{x}'_j(0))$$

for all $t, t' \geq 0$.

We assume that the system (2.4) is monotone. It means that displacement of the vehicle at time t is larger when larger inputs are applied and/or larger initial states are considered. The standard longitudinal dynamical model (2.5) satisfies the above assumption [30].

2.3 Supervisor Design and Safety Verification

The objective of this paper is to design a supervisor that returns an input signal that makes the system's state avoid the bad set. The supervisor is a map

$$s : (\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{u}_d) \mapsto \mathbf{u},$$

where $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ is the current state of vehicles, $\mathbf{u}_d \in \mathbf{U}$ is the vector of desired inputs applied by drivers, and $\mathbf{u} \in \mathcal{U}$ is an input signal returned by the supervisor. The current state and desired input are measured through cameras, lidars, and radars on the roadside or through each vehicle's on-board sensors whose measurements can

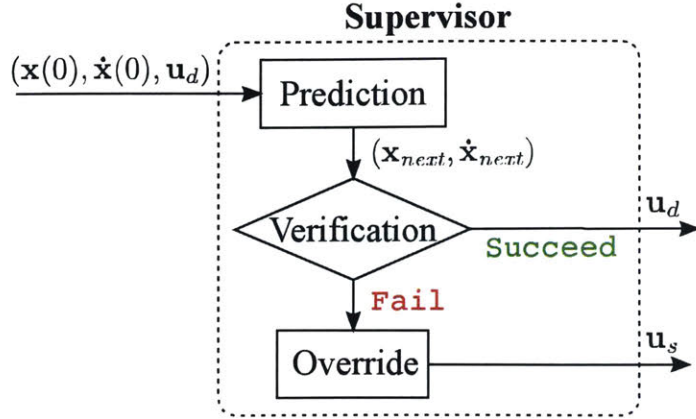


Figure 2-3: Supervisor returns the desired input \mathbf{u}_d if the input will not cause a future collision, and otherwise returns a safe input \mathbf{u}_s .

then be communicated to the supervisor. If the desired input \mathbf{u}_d will not cause a future collision, the supervisor returns $\mathbf{u} = \mathbf{u}_d$. Otherwise, the supervisor neglects the desired input and returns $\mathbf{u} = \mathbf{u}_s$ to prevent the predicted collision, where $\mathbf{u}_s \in \mathcal{U}$ is an input signal that keeps the system's state, starting from $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, outside the bad set B . The supervisor is implemented as a discrete-time algorithm with time step τ , which is typically $\tau = 0.1$ s for intelligent transportation applications [60].

The most challenging part in the design of the supervisor is to determine whether the desired input \mathbf{u}_d will cause a future collision, that is, to determine whether the supervisor has to change the mode to the OVERRIDE mode (see Figure 1-1(b)). To do this, we solve a *safety verification problem*.

Problem 2.1 (Safety verification). Given a state $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$, determine whether

$$\exists \mathbf{u} \in \mathcal{U} : \mathbf{x}(t, \mathbf{u}, \mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}}) \notin B, \forall t \geq 0. \quad (2.7)$$

In other words, the safety verification problem is the problem of determining whether a given state is inside the backward reachable set of the bad set (defined as the set of starting states that will reach the bad set regardless of control input). A state outside the backward reachable set is a solution to Problem 2.1.

Given a state $(\mathbf{x}(\tau, \mathbf{u}_d), \dot{\mathbf{x}}(\tau, \mathbf{u}_d))$, one-step ahead state with the desired input from the current state, if Problem 2.1 has a feasible solution, then the desired input \mathbf{u}_d

will not cause an unavoidable collision. Otherwise, the desired input \mathbf{u}_d must be overridden to avoid a predicted future collision. To sum up, the supervisor s at time $t = k\tau$ is designed as follows:

$$s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d) = \begin{cases} \text{if } \exists \mathbf{u} \in \mathcal{U} : \mathbf{x}(t, \mathbf{u}, \mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}}) \notin B, \forall t \geq 0, \\ \mathbf{u}_d & \text{where } \mathbf{x}_{\text{next}} = \mathbf{x}(\tau, \mathbf{u}_d, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \\ & \dot{\mathbf{x}}_{\text{next}} = \dot{\mathbf{x}}(\tau, \mathbf{u}_d, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \\ \mathbf{u}_s & \text{otherwise.} \end{cases} \quad (2.8)$$

where \mathbf{u}_s is a safe input signal defined on time $[0, \tau)$ that satisfies

$$\exists \mathbf{u} \in \mathcal{U} : \mathbf{x}(t, \mathbf{u}, \mathbf{x}(\tau, \mathbf{u}_s, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \dot{\mathbf{x}}(\tau, \mathbf{u}_s, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))) \notin B, \forall t \geq 0.$$

By construction, the supervisor s guarantees safety because it always returns the input that makes the one-step ahead state outside the backward reachable set of the bad set. Also, the supervisor s is *least restrictive*, that is, it returns the desired input \mathbf{u}_d if and only if the desired input will not cause the situation where no control input exists to avoid the bad set.

The supervisor is said to be *nonblocking* if

$$s(\mathbf{x}((k-1)\tau), \dot{\mathbf{x}}((k-1)\tau), \mathbf{u}_d) \neq \emptyset \implies s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d) \neq \emptyset,$$

which means by induction that if we have initially $s(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{u}_d) \neq \emptyset$, then the supervisor always returns a nonempty output.

Chapter 3

Preliminary Results

Our approach to solving the safety verification problem (Problem 2.1) is to translate it into a *scheduling* problem. In this section, we introduce fundamentals of scheduling problems, and explain a concept of equivalence between two decision problems. We also show an example of a scheduling problem that is equivalent to the safety verification problem in a simple intersection model.

3.1 Scheduling

Scheduling problems are described by jobs, machines, and processing characteristics [18, p. 4]. Jobs represent tasks which have to be executed; machines represent scarce resources such as facilities where jobs are processed; and processing characteristics include release times, deadlines, and process times. Scheduling problems are the problems of assigning jobs to machines at particular times. A scheduling problem is called *single-machine scheduling* if there is only one machine, and is called *jobshop scheduling* if there are more than one machines involved and each job follows its own predetermined routes of machines.

3.1.1 Single-machine Scheduling

Given a set of n jobs, an operation i , which refers to the process of job i on a single machine, is associated with release time r_i , deadline d_i , and process time p_i . Scheduling jobs means finding a schedule $\mathbf{t} = (t_1, \dots, t_n) \in \mathbb{R}^n$ that satisfies

$$\begin{cases} r_i \leq t_i \leq d_i - p_i, \\ (t_i, t_i + p_i) \cap (t_j, t_j + p_j) = \emptyset, \quad \forall i \neq j. \end{cases} \quad (3.1)$$

A feasible schedule must satisfy that each operation starts after the release time and is completed before the deadline, and only one job is processed on the machine at a time.

Scheduling problems are usually formulated as optimization problems. For example, jobs are scheduled to minimize the *maximum lateness* [50]

$$L_{\max} := \max_{i \in \{1, 2, \dots, n\}} (t_i + p_i - d_i), \quad (3.2)$$

or to minimize the maximum completion time

$$C_{\max} := \max_{i \in \{1, 2, \dots, n\}} (t_i + p_i).$$

In Chapter 5, we use an objective function similar to the maximum lateness to check the possibility of scheduling jobs before deadlines.

Although, in general, single-machine scheduling problems are known to be NP-hard [50], the problem that minimizes the maximum completion time can be solved in polynomial time if $p_i = 1$ for all i [27,53]. We adopt this result to address the complexity issue of the safety verification problem in Chapter 4.

There is another class of scheduling problems called *inserted idle-time (IIT) scheduling*. An inserted idle-time is an open time interval when the machine is deliberately held idle while at least one job is waiting to be processed [35]. If we let (\bar{r}, \bar{p}) be an

idle-time, a feasible schedule must satisfy (3.1) and an additional constraint

$$(t_i, t_i + p_i) \cap (\bar{r}, \bar{p}) = \emptyset. \quad (3.3)$$

In Chapter 4, we handle the presence of vehicles that are not controlled by the supervisor by introducing idle-times.

3.1.2 Jobshop Scheduling

In jobshop scheduling, given a set of n jobs and a set of m machines, each job has its own predetermined route of machines to follow. A jobshop scheduling problem can be described by a graph representation [50]. In the graph, a node is an operation (i, j) , i.e., a pair of a machine and a job, denoting that job j is processed by machine i . The collection of all nodes is denoted by \mathcal{N} . A conjunctive arc connects two nodes (i, j) and (i', j) if the two operations are on the route of job j , and a disjunctive arc connects two nodes (i, j) and (i, j') if the sequence of the two operations is undetermined on machine i . Let \mathcal{C} denote a set of conjunctive arcs and \mathcal{D} a set of disjunctive arcs.

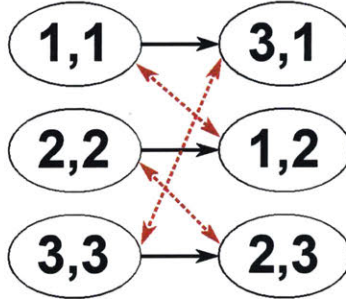


Figure 3-1: Example of the graph representation.

For example, in Figure 3-1, the black solid arrows are conjunctive arcs and the red dotted arrows are disjunctive arcs. We have

$$\mathcal{N} = \{(1, 1), (3, 1), (2, 2), (1, 2), (3, 3), (2, 3)\},$$

$$\mathcal{C} = \{(1, 1) \rightarrow (3, 1), (2, 2) \rightarrow (1, 2), (3, 3) \rightarrow (2, 3)\},$$

$$\mathcal{D} = \{(1, 1) \leftrightarrow (1, 2), (2, 2) \leftrightarrow (2, 3), (3, 3) \leftrightarrow (3, 1)\}.$$

The conjunctive arc $(1, 1) \rightarrow (3, 1)$ represents that job 1 should be processed by machine 1 and then machine 3. The disjunctive arc $(1, 1) \leftrightarrow (1, 2)$ represents that by machine 1, jobs 1 and 2 should be processed and the sequence is undetermined.

Given release time $r_{i,j}$, deadline $d_{i,j}$, and process time $p_{i,j}$ of an operation (i, j) , scheduling jobs means finding a schedule $\mathbf{t} = (t_{i,j} : (i, j) \in \mathcal{N})$ that satisfies

$$\begin{cases} r_{i,j} \leq t_{i,j} \leq d_{i,j} - p_{i,j}, \\ (t_{i,j}, t_{i,j} + p_{i,j}) \cap (t_{i,j'}, t_{i,j'} + p_{i,j'}) = \emptyset, \quad \forall (i, j) \leftrightarrow (i, j') \in \mathcal{D}. \end{cases} \quad (3.4)$$

3.2 Equivalence

We introduce a concept of equivalence [19] to describe the relation between two decision problems. In general, an instance of a problem is the information required to determine a solution to the problem. We say an instance is *accepted* by a problem if the solution to the problem is *yes* given the instance. A problem P_1 is reducible to a problem P_2 if for every instance I_1 of P_1 , an instance I_2 of P_2 can be constructed in polynomially bounded time such that P_2 accepts I_2 if and only if P_1 accepts I_1 . We say P_1 is *equivalent* to P_2 if and only if P_1 is reducible to P_2 and P_2 is reducible to P_1 .

3.3 Example

As an example, we consider that n vehicles move toward an intersection modeled as a single conflict area. The conflict area is located at (α_i, β_i) on the path of vehicle i . Given a state $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$, where $x_{\text{next},i}$ and $\dot{x}_{\text{next},i}$ denote the i -th entries of \mathbf{x}_{next}

and $\dot{\mathbf{x}}_{\text{next}}$, respectively, let

$$R_i := \min_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, x_{\text{next},i}, \dot{x}_{\text{next},i}) = \alpha_i\},$$

$$D_i := \max_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, x_{\text{next},i}, \dot{x}_{\text{next},i}) = \alpha_i\},$$

$$P_i(T_i) := \min_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, x_{\text{next},i}, \dot{x}_{\text{next},i}) = \beta_i$$

with constraint $x_i(T_i, u_i, x_{\text{next},i}, \dot{x}_{\text{next},i}) = \alpha_i\}$.

We formulate a scheduling problem, by considering vehicles as jobs and the conflict area as a single machine, to determine if scheduling times at which vehicles enter the conflict area is feasible.

Problem 3.1. Given a state $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$, determine if there exists a schedule $\mathbf{T} = (T_1, \dots, T_n) \subset \mathbb{R}^n$ that satisfies

$$\begin{cases} R_i \leq T_i \leq D_i, \\ (T_i, P_i(T_i)) \cap (T_j, P_j(T_j)) = \emptyset, \forall i \neq j. \end{cases} \quad (3.5)$$

A schedule T_i is the time when vehicle i is allowed to enter the conflict area. The entering time must be bounded by R_i and D_i , and two vehicles never meet inside the conflict area. The comparison between (3.5) and (3.1) shows that R_i functions as a release time, D_i as a deadline, and $P_i(T_i)$ as a process time.

It is proved in [16] that Problem 3.1 is equivalent to the safety verification problem (Problem 2.1). That is, if $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$ is accepted by Problem 3.1, it is also accepted by Problem 2.1, and *vice versa*. Note that by determining the existence of a feasible schedule (real numbers), we can determine the existence of a set of input signals satisfying (2.7) (functions of time).

Chapter 4

Supervisors at Single Conflict Area

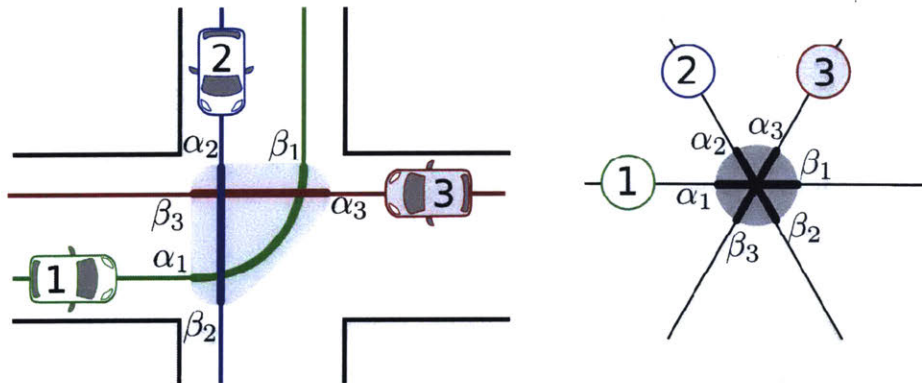


Figure 4-1: Road intersection (left) and its simplified model (right). An intersection is modeled as a conflict area (the shaded area), and its location is denoted by an open interval (α_i, β_i) along the path of car i . The supervisor communicates with and controls only cars 1 and 2 to avoid a collision among all three cars.

In this section, we model an intersection as a single conflict area as illustrated in Figure 4-1, and design supervisors that ensure safety inside the conflict area. In order to enable practical implementation of supervisors, we consider several uncertainty sources, such as modeling errors, measurement noises, and the presence of vehicles that are not equipped with communication modules and thus not controlled by the supervisors. In the nondeterministic setting, we design a least restrictive supervisor and a bit restrictive, but computationally efficient, supervisor that are robust to bounded uncertainties. We validate the supervisors via experiments using three in-scale cars and via computer simulations.

4.1 Uncertainty Sources

Before autonomous vehicles supersede the existing transportation paradigm, we expect road traffic to be composed of a mix of (i) autonomous vehicles, (ii) human-driven vehicles capable of communication with other vehicles and infrastructure, and of autonomous actions under specific conditions, and (iii) vehicles without any autonomous capability. We call (i) and (ii) *controlled* vehicles, and call (iii) *uncontrolled* vehicles, from the supervisor's perspective.

In the simulations and experiments presented in the end of this chapter, we model the longitudinal dynamics of the vehicles as a second-order affine system with disturbances:

$$\begin{aligned} \dot{x}_i &= v_i + d_{x,i}, \\ \dot{v}_i &= \begin{cases} \max(0, a_i v_i + b_i + \phi_i u_i + d_{v,i}) & \text{if } (v_i = v_{i,\min}), \\ \min(v_{i,\max}, a_i v_i + b_i + \phi_i u_i + d_{v,i}) & \text{if } (v_i = v_{i,\max}), \\ a_i v_i + b_i + \phi_i u_i + d_{v,i} & \text{otherwise.} \end{cases} \end{aligned} \quad (4.1)$$

The variable x_i is the position of vehicle i along its path, and v_i is the velocity of vehicle i , with velocity v_i bounded in the interval $[v_{i,\min}, v_{i,\max}] \subset [0, +\infty)$; $u_i \in \mathcal{U}_i$ is the control input signal bounded between $[u_{i,\min}, u_{i,\max}]$ (i.e., $u_i(t) \in [u_{i,\min}(t), u_{i,\max}(t)]$ for all t); parameters a_i and b_i are a mass-normalized damping coefficient and friction; $d_{x,i}$ and $d_{v,i}$ are bounded disturbance, capturing the effect of imperfect path following, of the coupling of lateral and longitudinal dynamics, of the road slope, and of the engine's nonlinear behavior; ϕ_i is a gain dependent on the battery charge. Note that the speed along the path (\dot{x}_i) is not equal to the actual speed (v_i) due to the disturbance $d_{x,i}$.

In the following, we let

$$\mathbf{s}_i = (x_i, v_i)$$

denote the full state of vehicle i and

$$\mathbf{d}_i := (d_{x,i}, d_{v,i}) \in \mathcal{D}_i,$$

denote its disturbance input signal, bounded between $[\mathbf{d}_{i,\min}, \mathbf{d}_{i,\max}]$, that is, $d_{x,i}(t) \in [d_{x,i,\min}(t), d_{x,i,\max}(t)]$ and $d_{v,i}(t) \in [d_{v,i,\min}(t), d_{v,i,\max}(t)]$ for all $t \geq 0$. Uncontrolled vehicles have the same dynamics, but neither the input u_i nor \mathbf{d}_i is controllable.

We denote by $\mathbf{s}_i(t, u_i, \mathbf{d}_i, \mathbf{s}_i(0))$ the state of vehicle i reached after time t with input signal u_i and disturbance signal \mathbf{d}_i starting from an initial state $\mathbf{s}_i(0)$, and similarly, by $x_i(t, u_i, \mathbf{d}_i, \mathbf{s}_i(0))$ the position reached at time t .

The state measurement $\mathbf{s}_{m,i} = (x_{m,i}, v_{m,i})$ is subject to noise $\boldsymbol{\delta}_i := (\delta_{x,i}, \delta_{v,i})$ where $\delta_{x,i}$ and $\delta_{v,i}$ are bounded errors for the position measurement $x_{m,i}$ and speed measurement $v_{m,i}$, respectively. That is, the actual state $\mathbf{s}_i = (x_i, v_i)$ is

$$x_i = x_{m,i} + \delta_{x,i}, \quad v_i = v_{m,i} + \delta_{v,i}. \quad (4.2)$$

The measurement noise is bounded between $\boldsymbol{\delta}_{i,\min}$ and $\boldsymbol{\delta}_{i,\max}$. The state measurement for all controlled and uncontrolled vehicles is available, for example, through cameras, lidars, and radars on the roadside.

For the system involving n controlled and \bar{n} uncontrolled vehicles, the aggregate state is denoted by $\mathbf{s} \in \mathbb{R}^{2(n+\bar{n})}$, the aggregate disturbance signal by $\mathbf{d} \in \mathcal{D}$, and the aggregate control input signal by $\mathbf{u} \in \mathcal{U}$. We partition the entries of \mathbf{u} into \mathbf{u}_c and \mathbf{u}_{uc} where $\mathbf{u}_c \in \mathcal{U}_c$ is the set of control input signals for controlled vehicles, and $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$ is the set of control input signals for uncontrolled vehicles. The aggregate state measurement is denoted by $\mathbf{s}_m \in \mathbb{R}^{2(n+\bar{n})}$, and the aggregated measurement noise by $\boldsymbol{\delta}$.

In the nondeterministic setting, the bad set is

$$B := \{\mathbf{x} \in \mathbf{X} : x_i \in (\alpha_i, \beta_i) \text{ and } x_j \in (\alpha_j, \beta_j)\}$$

for some controlled vehicle i and controlled/uncontrolled vehicle j .

The bad set contains points corresponding to collisions in which at least one controlled vehicle is involved, so that the supervisor can prevent the system's state from entering the bad set by controlling the input of controlled vehicles, \mathbf{u}_c .

4.2 Supervisor Design

Due to the presence of uncertainty sources, we should revise the supervisor design and the safety verification problem given in Chapter 2. In this section, we detail the revised structure of the supervisor and safety verification problem.

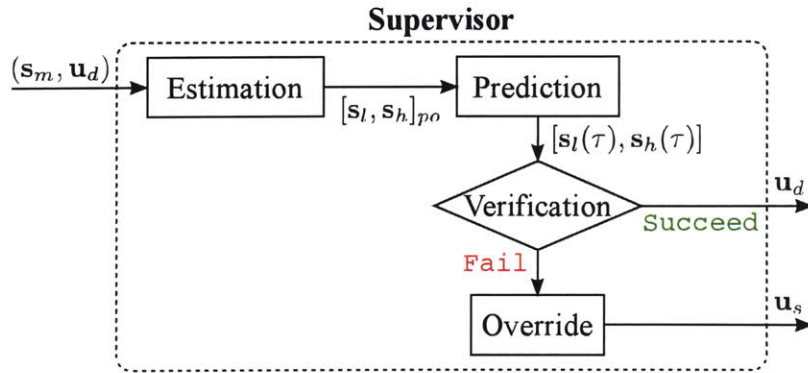


Figure 4-2: Structure of the supervisor. The verification algorithm predicts whether the supervisor should override controlled vehicles with a safe input \mathbf{u}_s to prevent collisions.

The robust supervisor consists of an estimation algorithm, a state prediction algorithm, a verification algorithm, and an override input algorithm as depicted in Figure 4-2. These run in a short time step of length τ to quickly adapt to the changing control inputs generated by drivers of controlled vehicles, and to the unmeasurable behavior of uncontrolled vehicles. We assume that the algorithms are synchronized among vehicles, and that communication takes a negligible time. Asynchrony and communication delays can be handled with little overhead [13].

4.2.1 Estimation

We call a *state estimate* the set of all states compatible with the system's measurements history at a given time. Consider the preorder $\mathbf{x} \leq \mathbf{y}$, meaning that each

element of \mathbf{x} is smaller than or equal to the corresponding element of \mathbf{y} . The dynamics of (4.1) are monotone in the control and disturbance inputs with respect to the above preorder [11]. This suggests to consider box state estimates, defined as the set

$$\{\mathbf{s} : \mathbf{s}_l \leq \mathbf{s} \leq \mathbf{s}_h\},$$

and identified by their lower and upper bound, $[\mathbf{s}_l, \mathbf{s}_h]$. At the current time $k\tau$, given a prior state estimate $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{pr}$ and the measured state $\mathbf{s}_m(k\tau)$ at the current time 0, the supervisor computes the posterior state estimate

$$[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po} := [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{pr} \cap [\mathbf{s}_m(k\tau) + \boldsymbol{\delta}_{\min}, \mathbf{s}_m(k\tau) + \boldsymbol{\delta}_{\max}], \quad (4.3)$$

where $\boldsymbol{\delta}_{\min}$ and $\boldsymbol{\delta}_{\max}$ are the vectors of measurement uncertainty bounds. Because the sets $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{pr}$ and $[\mathbf{s}_m(k\tau) + \boldsymbol{\delta}_{\min}, \mathbf{s}_m(k\tau) + \boldsymbol{\delta}_{\max}]$ contain the actual state, the posterior state estimate (4.3) is nonempty.

4.2.2 Prediction

Consider a prediction interval of length τ , equal to a supervisor time step. Given the current state estimate $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po}$, and the control input \mathbf{u}_d^k issued by the drivers of controlled vehicles at time $k\tau$, this algorithm predicts a box of possible future states $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$, exploiting monotonicity of the dynamics (4.1) with respect to input and disturbances to propagate the lower and upper bounds of the state estimate forward in time.

For controlled vehicle i , the control input $u_{d,i}^k$ in the interval $[0, \tau]$ is assumed to be constant, so the only uncertainty is due to the unknown disturbance input \mathbf{d}_i . The future state estimate is given by

$$\begin{aligned} [\mathbf{s}_{l,i}((k+1)\tau), \mathbf{s}_{h,i}((k+1)\tau)] := \\ [\mathbf{s}_i(\tau, u_{d,i}^k, \mathbf{d}_{i,\min}, (\mathbf{s}_{l,i}(k\tau))_{po}), \mathbf{s}_i(\tau, u_{d,i}^k, \mathbf{d}_{i,\max}, (\mathbf{s}_{h,i}(k\tau))_{po})]. \end{aligned} \quad (4.4)$$

Note that given a current state $\mathbf{s}(k\tau) \in [\mathbf{s}_{l,i}(k\tau), \mathbf{s}_{h,i}(k\tau)]_{po}$, the one-step ahead state

$\mathbf{s}(\tau, u_{d,i}^k, \mathbf{d}_i, \mathbf{s}(k\tau))$ must be bounded by the future state estimate (4.4) due to the monotonicity of the dynamics. For uncontrolled vehicle i , due to the lack of information about the control inputs, uncertainty is both on control and disturbance inputs. Therefore, the future state estimate is given by

$$\begin{aligned} & [\mathbf{s}_{l,i}((k+1)\tau), \mathbf{s}_{h,i}((k+1)\tau)] := \\ & [\mathbf{s}_{l,i}(\tau, u_{i,\min}, \mathbf{d}_{i,\min}, (\mathbf{s}_{l,i}(k\tau))_{po}), \mathbf{s}_{h,i}(\tau, u_{i,\max}, \mathbf{d}_{i,\max}, (\mathbf{s}_{h,i}(k\tau))_{po})]. \end{aligned}$$

The future state estimate is used as the prior state estimate at the next time step.

4.2.3 Verification

To account for the presence of uncertainty sources, we revise the safety verification problem given in Problem 2.1.

Problem 4.1 (Revised Safety Verification). Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, determine whether

$$\begin{aligned} \exists \mathbf{u}_c \in \mathcal{U}_c : \mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{s}_0) \notin B, \forall t \geq 0, \\ \forall \mathbf{u}_{uc} \in \mathcal{U}_{uc}, \forall \mathbf{d} \in \mathcal{D}, \forall \mathbf{s}_0 \in [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]. \end{aligned} \quad (4.5)$$

In other words, the safety verification problem in the nondeterministic setting is the problem of finding a control input signal of controlled vehicles that prevents the whole system's position from entering the bad set for all input signals of uncontrolled vehicles, for all disturbance input signals, and for all initial states inside a given state estimate.

In the following sections, we design discrete-time supervisors based on the solutions to Problem 4.1. The following supervisor s is said to be *least restrictive*:

$$s(\mathbf{s}_m(k\tau), \mathbf{u}_d^k) = \begin{cases} \mathbf{u}_d^k & \text{if } \exists \mathbf{u}_c \in \mathcal{U}_c : \mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{s}_0) \notin B, \forall t \geq 0, \\ & \forall \mathbf{u}_{uc} \in \mathcal{U}_{uc}, \forall \mathbf{d} \in \mathcal{D}, \forall \mathbf{s}_0 \in [\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)] \\ \mathbf{u}_s & \text{otherwise,} \end{cases} \quad (4.6)$$

in the sense that overrides are activated only when the desired input of controlled vehicles would lead to collisions at some future times for some disturbances and inputs of uncontrolled vehicles.

4.3 Least Restrictive Supervisor

In this section, we formulate an inserted idle-time (IIT) scheduling problem, which is solved straightforwardly, and prove that this problem is equivalent to Problem 4.1. Based on the solution to the IIT scheduling problem, we provide an algorithm implementing the least restrictive supervisor given in (4.6).

4.3.1 Inserted Idle-Time Scheduling Problem

We can translate Problem 4.1 into an inserted idle-time (IIT) scheduling problem, considering an intersection as a resource that all vehicles must share. A schedule represents the vector of times at which each controlled vehicle enters the intersection, and an inserted idle-time represents a set of time intervals during which uncontrolled vehicles occupy the intersection. This analogy is characterized mathematically using the concept of decision problem equivalence.

To formulate the IIT scheduling problem, we define processing characteristics.

Definition 4.1. Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, a release time R_i , a deadline D_i , and a process time $P_i(T_i)$ are defined for controlled vehicle i as follows.

If $x_{h,i}(\tau) < \alpha_i$,

$$R_i := \{t \geq 0 : x_i(t, u_{i,\max}, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i\},$$

$$D_i := \{t \geq 0 : x_i(t, u_{i,\min}, \mathbf{d}_{i,\min}, \mathbf{s}_{h,i}(\tau)) = \alpha_i\}.$$

Given a real number $T_i \geq 0$,

$$P_i(T_i) := \min_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i\}$$

$$\text{with constraint } x_i(T_i, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i\}.$$

If $x_{h,i}(\tau) \geq \alpha_i$, then set $R_i = D_i = 0$, and $P_i(T_i) = \{t : x_i(t, u_{i,\max}, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i\}$. If $x_{l,i}(\tau) \geq \beta_i$, then set $R_i = D_i = P_i(T_i) = 0$. If the constraint cannot be satisfied, set $P_i(T_i) = \infty$.

For uncontrolled vehicle i , an inserted idle-time, denoted by (\bar{R}_i, \bar{P}_i) , is defined as follows.

If $x_{h,i}(\tau) < \alpha_i$,

$$\bar{R}_i := \{t \geq 0 : x_i(t, u_{i,\max}, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i\},$$

$$\bar{P}_i := \{t \geq 0 : x_i(t, u_{i,\min}, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i\}.$$

If $x_{h,i}(\tau) \geq \alpha_i$, set $\bar{R}_i = 0$ and $\bar{P}_i = \{t : x_i(t, u_{i,\min}, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i\}$. If $x_{l,i}(\tau) \geq \beta_i$, set $\bar{R}_i = \bar{P}_i = 0$.

Due to the monotonicity, the release time R_i is the earliest time at which controlled vehicle i can enter the intersection (located at (α_i, β_i)) and the deadline D_i is the latest such time. Given that controlled vehicle i enters the intersection no earlier than T_i , process time $P_i(T_i)$ is the earliest time at which it can definitely exit the intersection. The inserted idle-time (\bar{R}_i, \bar{P}_i) is a time interval within which uncontrolled vehicle i can enter and exit the intersection regardless of its driver's input and disturbance signals.

The Inserted Idle-time (IIT) scheduling problem is formulated as follows.

Problem 4.2 (IIT scheduling). Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, determine if

there exists a schedule $\mathbf{T} = (T_1, \dots, T_n) \subset \mathbb{R}^n$ that satisfies

$$\begin{cases} R_i \leq T_i \leq D_i, \\ (T_i, P_i(T_i)) \cap (T_j, P_j(T_j)) = \emptyset, \\ (T_i, P_i(T_i)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset, \end{cases} \quad (4.7)$$

for all controlled vehicles $i \neq j$ and uncontrolled vehicle γ .

A schedule T_i indicates the time at which controlled vehicle i enters an intersection with $\mathbf{d}_{i,\max}$ from the state $\mathbf{s}_{h,i}(\tau)$, that is, $x_i(T_i, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i$ for some $u_i \in \mathcal{U}_i$. In Problem 4.2, the first constraint is induced on the schedule by the bounded input signals, and the second constraint implies that controlled vehicles i and j do not occupy the intersection simultaneously, thereby preventing a collision between controlled vehicles i and j . The last constraint implies that vehicle i does not occupy the intersection during the inserted idle-time, thereby preventing a collision between controlled vehicle i and uncontrolled vehicle γ . Thus, a schedule satisfying the above constraints is related to an input signal that satisfies (4.5). This is the essence of the proof of the following theorem. The proof can be found in Appendix A.

Theorem 4.1. *Problem 4.1 is equivalent to Problem 4.2.*

We now provide an algorithm that solves Problem 4.2, which, in turn, solves Problem 4.1 by Theorem 4.1. This algorithm contains two procedures. The first procedure, called SCHEDULING, assigns a schedule $\mathbf{T} = (T_1, \dots, T_n)$ to controlled vehicles according to a given sequence $\pi_0 \in \mathbb{R}^n$, that is, if $\pi_0 = (j_1, \dots, j_n)$, then $T_{j_1} \leq \dots \leq T_{j_n}$ which means that vehicle j_2 crosses the intersection after vehicle j_1 does. Also the procedure evaluates whether \mathbf{T} satisfies (4.7). The second procedure, called EXACT, inspects all possible sequences until it finds a sequence corresponding to a feasible schedule. The procedure returns *yes* if a feasible schedule is found and *no* otherwise.

The algorithm focuses on scheduling vehicles that are before the intersection (jobs that have not been processed). Given $\mathbf{s}_h(\tau)$, let $\mathcal{M} := \{i : x_{h,i}(\tau) < \alpha_i\}$ and \mathcal{P} be the

set of all permutations of \mathcal{M} . The cardinality of the set \mathcal{P} is $(|\mathcal{M}|)! = 1 \times 2 \times \dots \times |\mathcal{M}|$. Without loss of generality, we identify uncontrolled vehicles by integers $1, \dots, \bar{n}$ such that $\bar{R}_1 \leq \dots \leq \bar{R}_{\bar{n}}$.

Algorithm 4-1 Solution to Problem 4.2

```

1: procedure SCHEDULING( $\pi_0, [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$ )
2:    $\mathcal{M} = \{i \in \{1, 2, \dots, n\} : x_{h,i}(\tau) < \alpha_i\}$ 
3:    $T_i = 0, \forall i \notin \mathcal{M}$ 
4:    $P_{max} = \max_{i \notin \mathcal{M}} P_i(T_i)$ 
5:    $\pi = (j_1, j_2, \dots, j_{|\mathcal{M}|})$  where  $\pi_0 = (i_1, \dots, i_{|\pi_0|})$  and  $j_k = i_{|\pi_0| - |\mathcal{M}| + k}$ 
6:   for  $i = 1$  to  $|\mathcal{M}|$  do
7:     if  $i = 1$  then  $T_{j_1} = \max(R_{j_1}, P_{max})$ 
8:     else if  $i \geq 2$  then
9:        $T_{j_i} = \max(R_{j_i}, P_{j_{i-1}}(T_{j_{i-1}}))$ 
10:    for  $\gamma = 1$  to  $\bar{n}$  do
11:      if  $T_{j_i} \geq \bar{R}_\gamma$  then  $T_{j_i} = \max(T_{j_i}, \bar{P}_\gamma)$ 
12:      else if  $P_{j_i}(T_{j_i}) > \bar{R}_\gamma$  then  $T_{j_i} = \bar{P}_\gamma$ 
13:    if  $T_i \leq D_i$  for all  $i$  then return  $(\mathbf{T}, yes)$ 
14:    else return  $(\emptyset, no)$ 
15: procedure EXACT( $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$ )
16:   if  $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)] \cap B \neq \emptyset$  then return  $(\emptyset, no)$ 
17:   else
18:     if  $x_{h,i}(\tau) \geq \alpha_i$  for all  $i$  then return  $(\emptyset, yes)$ 
19:     else
20:       for all  $\pi \in \mathcal{P}$  do
21:          $(\mathbf{T}, ans) = \text{SCHEDULING}(\pi, [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)])$ 
22:         if  $ans = yes$  then return  $(\mathbf{T}, yes)$ 
23:       return  $(\emptyset, no)$ 

```

Procedure SCHEDULING in Algorithm 4-1 works as follows. In lines 2-4, if $x_{h,i}(\tau) \geq \alpha_i$, $T_i = 0$ because $R_i = D_i = 0$, and P_{max} is the time at which all of such vehicle

i exit the intersection. Given a vector of the indexes of vehicles π_0 , the assignment at line 5 extracts from π_0 the subvector π when $|\mathcal{M}| < |\pi_0|$. Given this sequence π , procedure SCHEDULING finds the earliest possible schedule. In a given sequence $\pi = (j_1, \dots, j_{|\mathcal{M}|})$, vehicle j_{i-1} crosses the intersection earlier than vehicle j_i (lines 7-9). In the **for** loop of lines 10-12, uncontrolled vehicle γ is considered. For an inserted idle-time $(\bar{R}_\gamma, \bar{P}_\gamma)$, either $T_{j_i} \geq \bar{P}_\gamma$ or $P_{j_i}(T_{j_i}) \leq \bar{R}_\gamma$ is true, thereby $(T_{j_i}, P_{j_i}(T_{j_i})) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. If $T_i \leq D_i$ for all i , then T_i satisfies (4.7), and thus, this procedure returns a feasible schedule in conjunction with the answer of *yes* (line 13). If $T_i > D_i$ for some i , since the schedule \mathbf{T} is constructed so that it takes the earliest possible value, there cannot be another schedule that satisfies $T_i \leq D_i$. Thus, no feasible schedule can be found according to this sequence π , and the procedure returns *no* (line 14).

Procedure EXACT in Algorithm 4-1 solves Problem 4.2 by inspecting all permutations in \mathcal{P} until a feasible schedule is found as noted in lines 20-22. In the worst case, all entries in \mathcal{P} must be evaluated, which means that the computation time increases exponentially as the number of controlled vehicles increases. Indeed, a scheduling problem is known to be NP-hard [16]. To avoid this computational complexity issue, we provide another algorithm that approximately solves the IIT scheduling problem with the guarantee of a quantified approximation bound. This approximate algorithm is provided in Section 4.4.

4.3.2 Least Restrictive Supervisor Algorithm

Based on the solution to the safety verification problem (Problem 4.1), we present an algorithm that implements the least restrictive supervisor.

Given a feasible schedule $\mathbf{T} = (T_1, \dots, T_n)$, we compute a corresponding input signal that satisfies (4.5). This is possible because the existence of a feasible schedule implies the existence of a safe input signal by Theorem 4.1. We define a safe input

generator $\sigma : ([\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)], \mathbf{T}) \mapsto \mathbf{u}_s$ as

$$u_{s,i} = \sigma_i([\mathbf{s}_{l,i}(\tau), \mathbf{s}_{h,i}(\tau)], T_i) \in \{u_i \in \mathcal{U}_i : x_i(P_i(T_i), u_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i \text{ and } x_i(T_i, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i\}, \quad (4.8)$$

if $T_i > 0$. Let $u_{s,i} = u_{i,\max}$ if $T_i = 0$. Here, $\sigma_i([\mathbf{s}_{l,i}(\tau), \mathbf{s}_{h,i}(\tau)], T_i)$ is the i -th entry of $\sigma([\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)], \mathbf{T})$. The safe input signal $u_{s,i}$ makes controlled vehicle i enter the intersection no earlier than T_i and exit it no later than $P_i(T_i)$.

The following algorithm implements the supervisor at time step k . When $k = 0$, let $[\mathbf{s}_l(0), \mathbf{s}_h(0)]_{pr} = \mathbb{R}^{2(n+\bar{n})}$.

Algorithm 4-2 Implementation of the least restrictive supervisor at time $k\tau$

- 1: **procedure** SUPERVISOR($\mathbf{s}_m(k\tau), \mathbf{u}_d^k$)
 - 2: **Estimate:** Given $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{pr}$ and $\mathbf{s}_m(k\tau)$, estimate $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po}$
 - 3: **Predict:** Given $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po}$ and \mathbf{u}_d^k , predict $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$
 - 4: $(\mathbf{T}_1, ans_1) = \text{EXACT}([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)])$
 - 5: **if** $ans_1 = \text{yes}$ **then**
 - 6: $\mathbf{u}_s^{k+1} = \sigma([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)], \mathbf{T}_1)$
 - 7: **return** \mathbf{u}_d^k
 - 8: **else**
 - 9: **Predict:** Given $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po}$ and \mathbf{u}_s^k , predict $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$
 - 10: $(\mathbf{T}_2, ans_2) = \text{EXACT}([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)])$
 - 11: $\mathbf{u}_s^{k+1} = \sigma([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)], \mathbf{T}_2)$
 - 12: **return** \mathbf{u}_s^k restricted to time $[0, \tau]$
 - 13: $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ is used as $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]_{pr}$
-

When the future state estimate with \mathbf{u}_d^k is accepted by the safety verification problem (line 5), the supervisor stores a safe input signal \mathbf{u}_s^{k+1} for possible uses at the next time step and returns \mathbf{u}_d^k (lines 6 and 7). Otherwise, the algorithm computes the future state estimate with \mathbf{u}_s^k , a safe input signal stored at the previous step (line 9). Using this new state estimate, the algorithm updates a safe input signal \mathbf{u}_s^{k+1} (line 11) and overwrites the desired input with a safe input signal \mathbf{u}_s^k restricted to time $[0, \tau]$

(line 12).

To initiate the procedure, we assume that the algorithm returns \mathbf{u}_d^0 when $k = 0$. Given this assumption, procedure SUPERVISOR in Algorithm 4-2 is nonblocking, which implies that it always returns a nonempty output, and ensures safety by implementing the supervisor given in (4.6). The proof of Theorem 4.2 is provided in Appendix A.

Theorem 4.2. *Procedure SUPERVISOR in Algorithm 4-2 implements the supervisor s designed in (4.6), and it is nonblocking.*

4.4 Approximate Supervisor

Even though a least restrictive supervisor is theoretically computable, exploring all possible sequences for a large number of vehicles is typically too demanding. Thus, we present a novel approach to finding a good candidate sequence, and the construction of an approximate supervisor.

4.4.1 Approximate Verification

As mentioned in Section 3.1.1, while scheduling problems on a single machine with arbitrary release times, deadlines, and process times are known to be NP-hard, the complexity can be reduced to $O(n \log n)$ if process times of all jobs are identical [27]. The basic idea of the scheduling algorithm in [27] relies on the concept of a “forbidden region,” a set of time intervals in which jobs are forbidden to be scheduled, and the “earliest deadline scheduling (EDD)” rule. The EDD rule is a rule that assigns the machine to jobs in the order of their deadlines. The algorithm computes forbidden regions and applies the EDD rule to solve the scheduling problem with identical process times.

We present Algorithm 4-3, which is a modified version of the result in [27] to handle inserted idle-times, and use it to compute a good candidate sequence of vehicles. We define initial forbidden regions \mathbf{F} to account for the idle-times and set them as inputs of Algorithm 4-3, whereas forbidden regions are initially declared empty in [27]. We

call \mathbf{r} and \mathbf{d} the vectors of r_i and d_i , and call σ a vector of indexes in the increasing order of r_i (so that $r_{\sigma_1} \leq r_{\sigma_2} \leq \dots$). The forbidden regions \mathbf{F} is the union of F_γ .

Algorithm 4-3 Modified version of the result of [27]

```

1: procedure COMPUTE SEQUENCE( $\mathbf{r}, \mathbf{d}, \mathbf{F}$ )
2:   Forbidden region declaration:
3:   for  $i = n$  to 1 do
4:     for  $j \in \{j : d_j \geq d_{\sigma_i}\}$  do
5:       if  $c_j$  undefined then  $c_j = d_j$  else  $c_j = c_j - 1$ 
6:       if  $c_j \in F_\gamma$  for some  $\gamma$  then  $c_j = \inf F_\gamma$ 
7:       if  $\sigma_i = 1$  or  $r_{\sigma_{i-1}} < r_{\sigma_i}$  then
8:          $c = \min\{c_j : c_j \text{ defined}\}$ 
9:         if  $c < r_{\sigma_i}$  then the problem is infeasible
10:        if  $r_{\sigma_i} \leq c < r_{\sigma_i} + 1$  then  $\mathbf{F} = \mathbf{F} \cup (c - 1, r_{\sigma_i})$ 
11:   Sequence computation:
12:    $s = 0, \mathcal{A} = \emptyset, \mathcal{B} = \{1, \dots, |\mathbf{R}|\}$ 
13:   while  $\mathcal{B} \neq \emptyset$  do
14:     if  $\mathcal{A} = \emptyset$  then  $s = \min_{i \in \mathcal{B}} r_i$ 
15:     if  $s \in F_\gamma$  for some  $\gamma$  then  $s = \sup F_\gamma$ 
16:      $\mathcal{A} = \{k \in \mathcal{B} : r_k \geq s\}$  and  $j = \arg \min_{i \in \mathcal{A}} d_i$ 
17:      $t_j = s, s = s + 1,$  and  $\mathcal{B} = \mathcal{B} \setminus \{j\}$ 
18:    $\pi^* :=$  a sequence of vehicles in increasing order of  $\mathbf{t} = (t_1, \dots, t_n)$ 
19:   return  $(\pi^*, \mathbf{t})$ 

```

Algorithm 4-3 is a straightforward variation of the scheduling algorithm proposed in [27], changed so as to return a sequence π^* even if the scheduling problem does not admit a feasible schedule \mathbf{t} . The feasibility of the scheduling problem of unit-length jobs is determined during the forbidden region declaration (line 9), while the feasibility result is not returned by procedure COMPUTE SEQUENCE. Sequence computation uses the EDD rule, that is, one with smallest d_i among the jobs is scheduled earlier

than the others while avoiding the forbidden region (lines 15-17). This algorithm is proved to yield the exact solution to the scheduling problem of unit-length jobs.

We assign to all vehicles a time interval of equal length to cross the intersection, and formulate the IIT scheduling problem with identical process times. The identical process time θ_{\max} is defined as

$$\theta_{\max} = \max_{i \in \{1, 2, \dots, n\}} \max_{R_i \leq T_i \leq D_i} (P_i(T_i) - T_i). \quad (4.9)$$

Here, θ_{\max} is the maximum time that any controlled vehicle spends crossing an intersection, so that $\theta_{\max} \geq P_i(T_i) - T_i$ for all i for all $T_i \in [R_i, D_i]$. In other words, all controlled vehicles are guaranteed to cross the intersection within θ_{\max} . By replacing $P_j(T_j)$ in Problem 4.2 with $T_j + \theta_{\max}$, we state the IIT scheduling problem with identical process times as follows.

Problem 4.3 (IIT scheduling with identical process times). Given the processing characteristics used in Problem 4.2, determine if there exists a schedule $\mathbf{T} = (T_1, \dots, T_n) \subset \mathbb{R}^n$ that satisfies

$$\begin{cases} R_i \leq T_i \leq D_i, \\ (T_i, T_i + \theta_{\max}) \cap (T_j, T_j + \theta_{\max}) = \emptyset, \\ (T_i, T_i + \theta_{\max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset, \end{cases} \quad (4.10)$$

for all controlled vehicles $i \neq j$ and uncontrolled vehicle γ if $T_i, T_j > 0$. If $T_i = 0$, $(0, P_i(0))$ replaces $(T_i, T_i + \theta_{\max})$.

We present an algorithm that solves Problem 4.3 by using Algorithm 4-3. For this, all the parameters need to be normalized by θ_{\max} because Algorithm 4-3 assumes unit process times. For notational simplicity, let $\mathcal{M} = \{i \in \{1, 2, \dots, n\} : x_{h,i}(\tau) < \alpha_i\} = \{1, 2, \dots, |\mathcal{M}|\}$, and $\mathbf{r} = (r_1, \dots, r_{|\mathcal{M}|})$, $\mathbf{d} = (d_1, \dots, d_{|\mathcal{M}|})$, and $\mathbf{F} = \cup_\gamma F_\gamma$.

Algorithm 4-4 Solution to Problem 4.3

```
1: procedure SOLUTION TO PROBLEM 4.3( $[s_l(\tau), s_h(\tau)]$ )
2:   if  $[x_l(\tau), x_h(\tau)] \cap B \neq \emptyset$  then return  $(\emptyset, no)$ 
3:   else
4:      $\mathcal{M} = \{i \in \{1, 2, \dots, n\} : x_{h,i}(\tau) < \alpha_i\}$ 
5:      $T_i = 0, \forall i \notin \mathcal{M}$ 
6:      $p_{\max} = \max_{i \notin \mathcal{M}} P_i(T_i) / \theta_{\max}$ 
7:      $r_i = \max(R_i / \theta_{\max}, p_{\max}), d_i = D_i / \theta_{\max}$  for all  $i \in \mathcal{M}$ 
8:      $F_\gamma = (\max(0, \bar{R}_\gamma / \theta_{\max} - 1), \bar{D}_\gamma / \theta_{\max})$  for all  $\gamma \in \{1, \dots, \bar{n}\}$ 
9:      $(\pi^*, \mathbf{t}) = \text{COMPUTE SEQUENCE}(\mathbf{r}, \mathbf{d}, \mathbf{F})$ 
10:     $T_i = t_i \theta_{\max}, \forall i \in \mathcal{M}$ 
11:    if  $T_i \leq D_i$  for all  $i$  then
12:      return  $(\mathbf{T}, yes)$ 
13:    else
14:      return  $(\mathbf{T}, no)$ 
```

Because procedure COMPUTE SEQUENCE in Algorithm 4-3 handles only vehicles before the intersection (i.e., jobs not yet processed), procedure SOLUTION TO PROBLEM 4.3 separately considers vehicles inside or after the intersection (lines 5 and 6) and vehicles before the intersection (lines 7-10). This procedure exactly solves Problem 4.3 (as proved in [27]).

Procedure SOLUTION TO PROBLEM 4.3 in Algorithm 4-4 can be used as an approximate solution to Problem 4.2 (thus Problem 4.1 by Theorem 4.1), because a feasible schedule of Problem 4.3 is also feasible in Problem 4.2. Note that Problem 4.3 assigns a time interval $(T_i, T_i + \theta_{\max})$ to each vehicle i while vehicle i actually stays inside the intersection during a time interval $(T_i, P_i(T_i))$ where $P_i(T_i) \leq T_i + \theta_{\max}$. Thus, if the supervisor is designed based on the solution given by Algorithm 4-4, it tries to ensure that the gap between the entering times of two controlled vehicles is at least θ_{\max} . This supervisor can be very conservative, and was presented in [14] without considering the presence of uncontrolled vehicles. To design a less restrictive

supervisor, we provide another approximate solution to Problem 4.1 in the following algorithm, which assigns the time interval $(T_i, P_i(T_i))$ to each vehicle i while considering the same sequence π^* used in Algorithm 4-4.

Algorithm 4-5 Approximate Solution of Problem 4.1

```

1: procedure APPROXIMATE( $[s_l(\tau), s_h(\tau)]$ )
2:   if  $[x_l(\tau), x_h(\tau)] \cap B \neq \emptyset$  then return  $(\emptyset, no)$ 
3:   else
4:      $\mathcal{M} = \{i \in \{1, 2, \dots, n\} : x_{h,i}(\tau) < \alpha_i\}$ 
5:      $T_i = 0, \forall i \notin \mathcal{M}$ 
6:      $p_{\max} = \max_{i \notin \mathcal{M}} P_i(T_i) / \theta_{\max}$ 
7:      $r_i = \max(R_i / \theta_{\max}, p_{\max}), d_i = D_i / \theta_{\max}$  for all  $i \in \mathcal{M}$ 
8:      $F_\gamma = (\bar{R}_\gamma / \theta_{\max} - 1, \bar{D}_\gamma / \theta_{\max})$  for all  $\gamma \in \{1, \dots, \bar{n}\}$ 
9:      $(\pi^*, \mathbf{t}) = \text{COMPUTE SEQUENCE}(\mathbf{r}, \mathbf{d}, \mathbf{F})$ 
10:     $(\mathbf{T}, ans) = \text{SCHEDULING}(\pi^*, [s_l(\tau), s_h(\tau)])$ 
11:    return  $(\mathbf{T}, ans)$ 

```

This procedure schedules vehicles according to a sequence returned by procedure COMPUTE SEQUENCE, thereby inheriting computational efficiency. The only difference between Algorithms 4-4 and 4-5 is that Algorithm 4-5 calls procedure SCHEDULING to find a schedule based on a sequence π^* , not using a schedule \mathbf{t} returned by COMPUTE SEQUENCE (lines 9 and 10).

We will prove that procedure APPROXIMATE (approximate solution to Problem 4.2) is more conservative than procedure EXACT (exact solution to Problem 4.2) because it computes a schedule based on one candidate sequence, not all possible sequences. In order to quantify the degree of conservatism, we prove two theorems. The first theorem states that if procedure APPROXIMATE returns *yes*, then there exists an input signal to avoid the bad set. In the second theorem, if procedure APPROXIMATE returns *no*, then there does not exist an input signal to avoid an *inflated bad set*, which accounts for the conservatism. The proofs of these theorems are provided in Appendix A.

Theorem 4.3. *If APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) returns yes, then EXACT($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) also returns yes (i.e., there is an input signal satisfying (4.5)).*

However, the converse of Theorem 4.3 is not true. That is, for some instances accepted by Problem 4.1, procedure APPROXIMATE returns *no*. To consider such instances, we introduce an *inflated* bad set.

Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, a feasible schedule $\mathbf{T} = (T_1, \dots, T_n)$ in Problem 4.2 satisfies for some $u_i \in \mathcal{U}_i$,

$$x_i(T_i, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i, \quad x_i(P_i(T_i), u_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i,$$

that is, vehicle i enters the intersection no earlier than T_i and exits no later than $P_i(T_i)$. In contrast, a feasible schedule in Problem 4.3 satisfies for some $u_i \in \mathcal{U}_i$,

$$x_i(T_i, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i, \quad x_i(T_i + \theta_{\max}, u_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) \geq \beta_i,$$

Given that the farthest distance that controlled vehicle i can travel during θ_{\max} is $\theta_{\max}v_{i,\max}$, we define an inflated intersection $(\alpha_i, \hat{\beta}_i)$ such that

$$\hat{\beta}_i := \alpha_i + \theta_{\max}v_{i,\max}.$$

Note that $\beta_i \leq \hat{\beta}_i$. Because the process times are only defined for controlled vehicles, $\hat{\beta}_\gamma = \beta_\gamma$ for all uncontrolled vehicle γ . Thus, the inflated bad set \hat{B} is defined as follows:

$$\hat{B} := \{\mathbf{x} \in \mathbf{X} : x_i \in (\alpha_i, \hat{\beta}_i) \text{ and } x_j \in (\alpha_j, \hat{\beta}_j)\}$$

for some controlled vehicle i and controlled/uncontrolled vehicle j .

By replacing the bad set B in Problem 4.1 with the inflated bad set \hat{B} , we can formulate the relaxed verification problem. The following theorem is a key result that shows the approximation bound of procedure APPROXIMATE. The proof can be found in Appendix A.

Theorem 4.4. *If $\text{APPROXIMATE}([\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]) = (\emptyset, no)$, then there is no input signal $\mathbf{u}_c \in \mathcal{U}_c$ that guarantees $\mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{s}_0) \notin \hat{B}$ for all $t \geq 0$ for all $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, and $\mathbf{s}_0 \in [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$.*

In summary, Theorem 4.3 states that if procedure APPROXIMATE returns *yes*, there exists an input signal to avoid the bad set B for all uncertainties. Theorem 4.4 states that if the procedure returns *no*, there does not exist an input signal to avoid the inflated bad set \hat{B} for all uncertainties. Thus, \hat{B} represents the approximation bound of the solution given by procedure APPROXIMATE.

The approximate solution presented in this section is computationally efficient because it evaluates only one sequence of vehicles, as opposed to the exact solution given in Section 4.3 which evaluates all possible sequences. We can adopt other procedures that efficiently generate a candidate sequence. For example, Problem 4.3 can be written as a mixed integer linear programming (MILP) problem, which can be solved quickly by available software such as CPLEX [33], and the solution to Problem 4.3 can be used to extract a good candidate sequence.

4.4.2 Approximate Supervisor Algorithm

In order for a supervisor to run in real-time, the safety verification problem must be solved within a time step τ . However, if a large number of controlled vehicles are involved, the problem becomes intractable. Thus, we design an approximate, but computationally efficient, supervisor by employing the results in Section 4.4.1.

Here, we present an algorithm implementing an approximate supervisor by using procedure APPROXIMATE to verify whether or not to override drivers.

Algorithm 4-6 Implementation of an approximate supervisor

```
1: procedure APPROXIMATESUPERVISOR( $s_m(k\tau), \mathbf{u}_d^k$ )
2:   Estimate: Given  $[s_l(k\tau), s_h(k\tau)]_{pr}$  and  $s_m(k\tau)$ , estimate  $[s_l(k\tau), s_h(k\tau)]_{po}$ 
3:   Predict: Given  $[s_l(k\tau), s_h(k\tau)]_{po}$  and  $\mathbf{u}_d^k$ , predict  $[s_l((k+1)\tau), s_h((k+1)\tau)]$ 
4:    $(\mathbf{T}_1, ans_1) = \text{APPROXIMATE}([s_l((k+1)\tau), s_h((k+1)\tau)])$ 
5:   if  $ans_1 = \text{yes}$  then
6:      $\pi^k =$  a vector of indexes in the increasing order of  $\mathbf{T}_1$ 
7:      $\mathbf{u}_s^{k+1} = \sigma([s_l((k+1)\tau), s_h((k+1)\tau)], \mathbf{T}_1)$ 
8:     return  $\mathbf{u}_d^k$ 
9:   else
10:    Predict: Given  $[s_l(k\tau), s_h(k\tau)]_{po}$  and  $\mathbf{u}_s^k$ , predict  $[s_l((k+1)\tau), s_h((k+1)\tau)]$ 
11:     $(\mathbf{T}_2, ans_2) = \text{APPROXIMATE}([s_l((k+1)\tau), s_h((k+1)\tau)])$ 
12:    if  $ans_2 = \text{no}$  then
13:       $(\mathbf{T}_2, ans_3) = \text{SCHEDULING}(\pi^{k-1}, [s_l((k+1)\tau), s_h((k+1)\tau)])$ 
14:       $\pi^k =$  a vector of indexes in the increasing order of  $\mathbf{T}_2$ 
15:       $\mathbf{u}_s^{k+1} = \sigma([s_l((k+1)\tau), s_h((k+1)\tau)], \mathbf{T}_2)$ 
16:      return  $\mathbf{u}_s^k$  restricted to time  $[0, \tau]$ 
17:     $[s_l((k+1)\tau), s_h((k+1)\tau)]$  is used as  $[s_l((k+1)\tau), s_h((k+1)\tau)]_{pr}$ 
```

This procedure stores a feasible sequence of vehicles crossing the intersection at every time step (lines 6 and 14) such that $\pi^k = (j_1, \dots, j_n)$ implies $T_{j_1} \leq \dots \leq T_{j_n}$. These steps are necessary because when a sequence considered in line 11 does not yield a feasible schedule, the previous step's sequence π^{k-1} can be used to generate a feasible solution (line 13). The fact that ans_3 is always *yes* is proved in Theorem 4.5. In [14], the approximate supervisor does not store a sequence at each time and uses SOLUTION TO PROBLEM 4.3 in place of APPROXIMATE. As a result, their approximate supervisor cannot find a feasible schedule at every time step and considers the maximum process time, θ_{max} . In contrast, our approximate supervisor always finds a feasible schedule based on the most current state estimation and considers process times $P_i(T_i)$ for all controlled vehicles i , thereby being less conservative.

The computation time of APPROXIMATE and SCHEDULING determines the computation time of APPROXIMATESUPERVISOR. Thus, Algorithm 4-6 has polynomial computational complexity. Also, this approximate supervisor is more restrictive than the least restrictive supervisor s given in (4.6) in the sense that it overrides controlled vehicles more frequently than s . This implies that the approximate supervisor guarantees that the system's position never enters the bad set B .

Theorem 4.5. *Procedure APPROXIMATESUPERVISOR in Algorithm 4-6 is more restrictive than the supervisor s given by (4.6), that is,*

$$\text{APPROXIMATESUPERVISOR}(s_m(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k \implies s(s_m(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k.$$

Moreover, Algorithm 4-6 is nonblocking.

The proof of Theorem 4.5 can be found in Appendix A.

4.5 Validation of the Supervisors

We present the validation results of the supervisors given in Algorithm 4-2 and Algorithm 4-6 via computer simulations and experiments.

4.5.1 Computer Simulations

In this section, we present the results of computer simulations of Algorithm 4-2 (least restrictive supervisor) and Algorithm 4-6 (approximate supervisor). The simulations were performed using MATLAB on a personal computer with an 3.10 GHz Intel Core i7-3770s processor with 8 GB RAM, with control gain $\phi_i = 1$ and desired input $u_{d,i} = 150$ for all i . The values of the other parameters are presented in the next section.

In Figure 4-3, we show a run of Algorithm 4-2 and Algorithm 4-6 on the same initial conditions, desired input signals, and disturbance signals in a scenario with eight controlled vehicles and two uncontrolled vehicles. The supervisor overrides the controlled vehicles (when the blue bands appear) to avoid a collision at the intersection

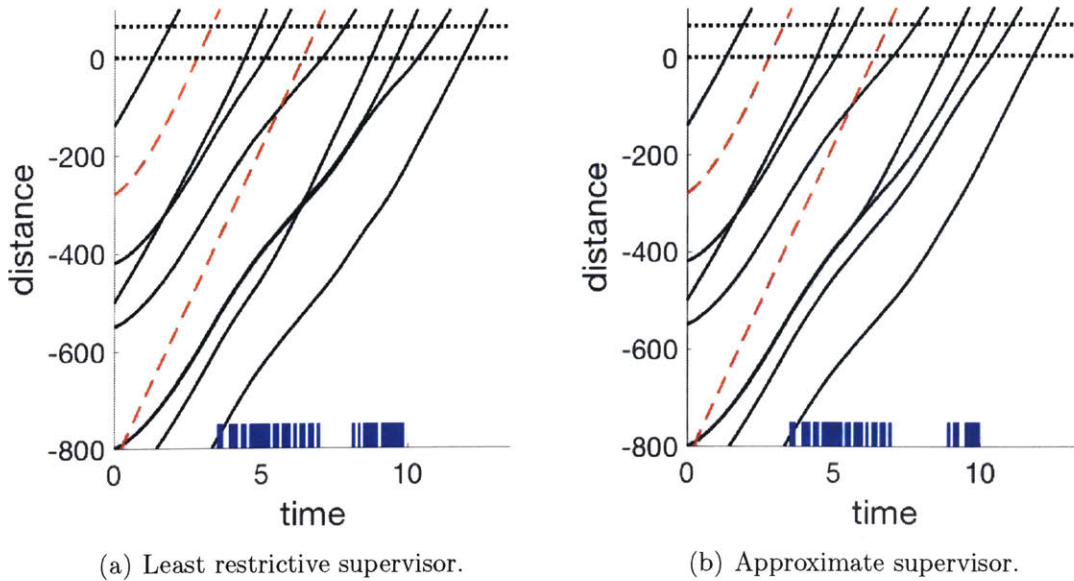


Figure 4-3: In simulation, the approximate supervisor intervenes no earlier than the least restrictive supervisor. The solid black and dotted red lines represent the actual position of the controlled and uncontrolled vehicles, respectively, with gray surrounding representing the state estimate. The blue bands on the bottom appear when the supervisor overrides the controlled vehicles to prevent a collision at the intersection at $(0, 65)$.

at $(0, 65)$. Notice that the approximate supervisor overrides the controlled vehicles no earlier than the least restrictive supervisor. That is, the approximate supervisor in practice performs close to the least restrictive supervisor. Algorithm 4-2 took 2.92 s per iteration while each iteration of Algorithm 4-6 was executed in less than 0.007 s.

To compare Algorithm 4-6 with the algorithm in [14], we have tested a scenario with four controlled vehicles and no uncontrolled vehicle (which the algorithm in [14] could not handle), running the two algorithms with the same initial conditions, desired input signals, and disturbance signals. The blue bands at the bottom of Figure 4-4 indicate times when the supervisor is overriding the drivers' desired input; the supervisor from [14], in the left panel, overrides much earlier than Algorithm 4-6, in the right panel. We also performed 100 tests with random initial conditions of four vehicles. The average ratio of the number of overrides to the number of steps until all vehicles exit the intersection (set to be constant at 200 steps) is 0.42 for the algorithm in [14] and 0.09 for Algorithm 4-6. This confirms that Algorithm 4-6 indeed restricts

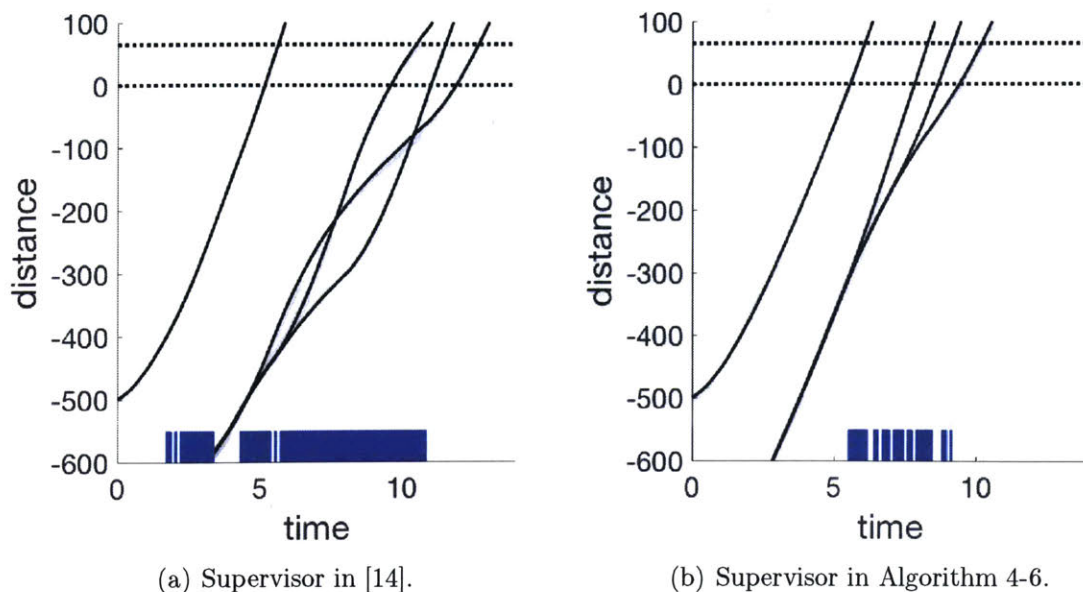


Figure 4-4: The blue bands at the bottom show that our supervisor in (b) starts an override later than the supervisor in (a). The same coloring is used as in Fig. 4-3

the drivers less than the supervisor in [14].

4.5.2 Experiments

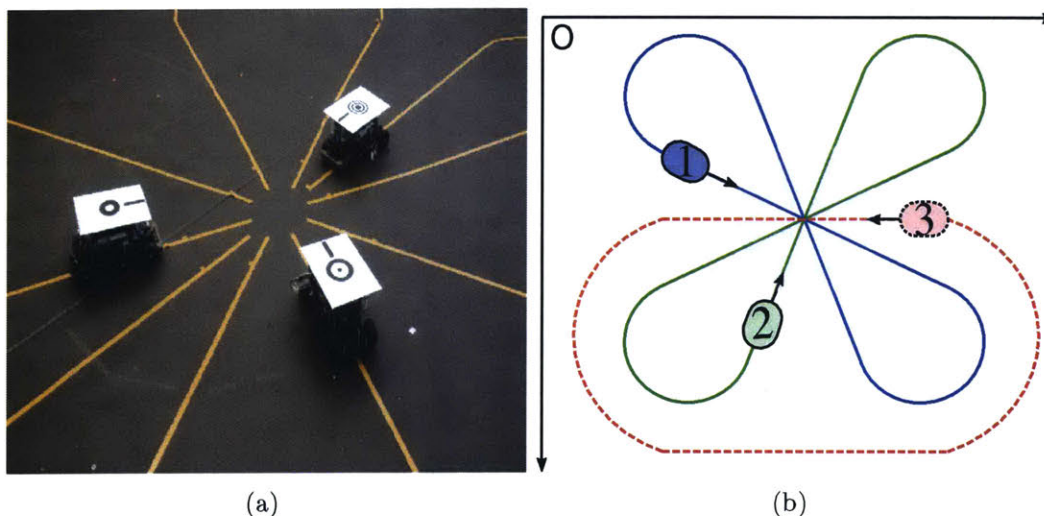


Figure 4-5: Experimental setup. The supervisor modifies the inputs of cars 1 and 2 only when necessary to avoid collisions at the intersection (shaded circle) in the presence of an uncontrolled vehicle (car 3).

We have implemented Algorithm 4-2 (least restrictive supervisor) in an intersec-

tion testbed. As shown in Figure 4-5(a), we use three in-scale cars. Cars 1 and 2 are controlled by an on-board computer and communicate with the supervisor via wifi. Car 3 is uncontrollable, and is remotely controlled by a human operator. The positions of all cars are measured by a camera system, while velocities of the controlled cars are measured through encoders. Each car follows one of the the three paths in Figure 4-5(b), which intersect at the center.

We have investigated the sources of disturbances $\mathbf{d}_i = (d_{x,i}, d_{v,i})$ for each car i and designed a compensating input to reduce their effects. For the disturbance input on the position dynamics, $d_{x,i} = d_{\text{proj},i}$, where $d_{\text{proj},i}$ is the effect of the car's wiggling motion about its nominal path on the longitudinal velocity. For the disturbance input on the speed dynamics,

$$d_{v,i} := d_{\text{power},i} + d_{\text{slope},i} + d_{\text{steer},i}$$

where $d_{\text{power},i}$, $d_{\text{slope},i}$, and $d_{\text{steer},i}$ are the effects on the longitudinal acceleration of the engine's nonlinear behavior, of the road slope, and of the lateral-longitudinal dynamics coupling, respectively. We model $d_{\text{power},i} = g_i e^{-t/h_i} u_i$ where g_i, h_i are a gain and a time constant, and u_i is the motor input. The disturbances $d_{\text{slope},i}$ and $d_{\text{steer},i}$ vary on the position x_i and are estimated off-line. The effects of $d_{\text{steer},i}$, $d_{\text{slope},i}$, and $d_{\text{power},i}$ are reduced through the introduction of a compensating input $c_i(t, x_i)$

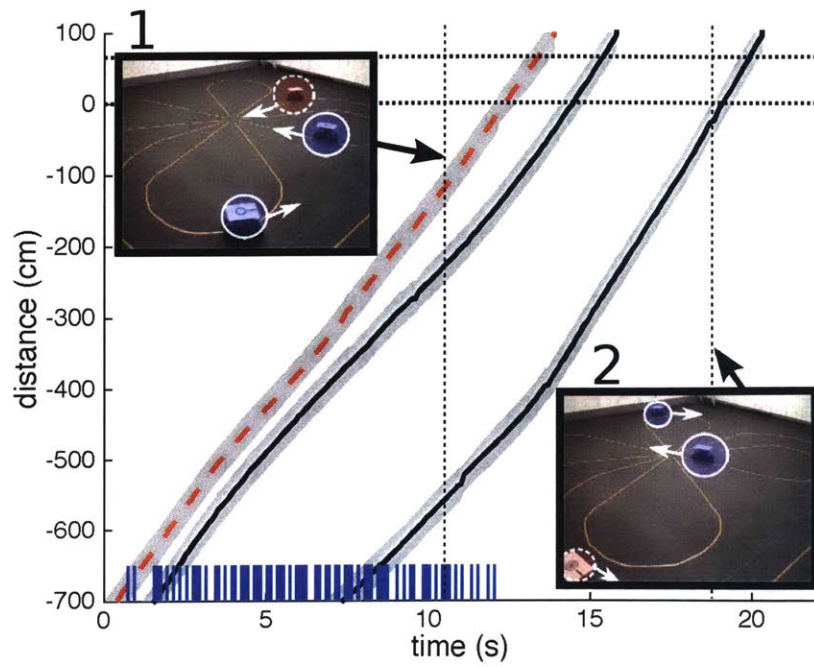
$$c_i(t, x_i) = - \frac{g_i e^{-t/h_i} + d_{\text{slope},i}(x_i) + d_{\text{steer},i}(x_i)}{g_i e^{-t/h_i} + \phi_i}.$$

By applying an input $u_i = u'_i + c_i(t, x_i)$, the car dynamics become

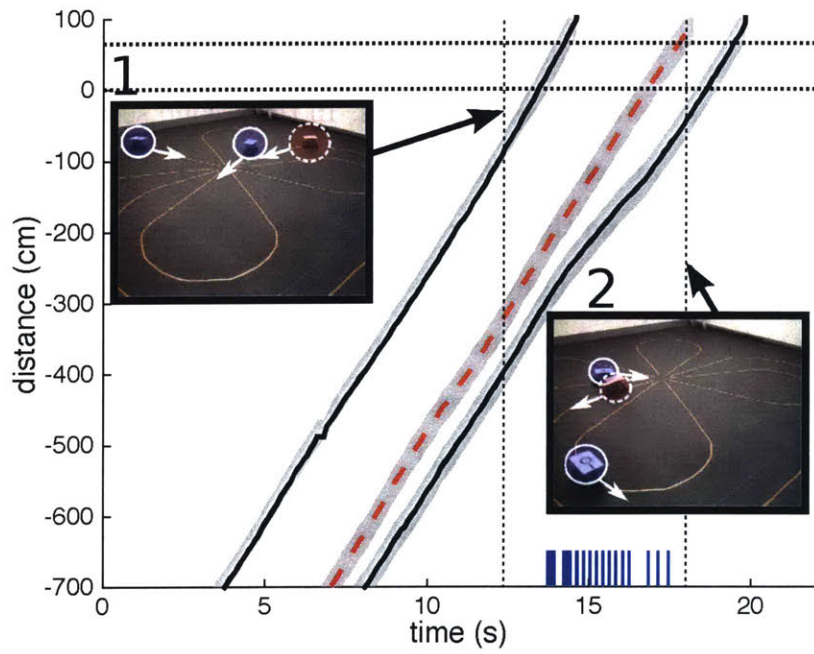
$$\dot{v}_i = a_i v_i + b_i + \phi_i u'_i + d'_{v,i}$$

where $d'_{v,i} = g_i e^{-t/h_i} (u'_i - 1)$ is the effective disturbance after the compensation. The gain ϕ_i in our experiments depends on the battery charge and is estimated on-line using adaptive control.

Through a set of experiments on each car, we have identified the values for the



(a)



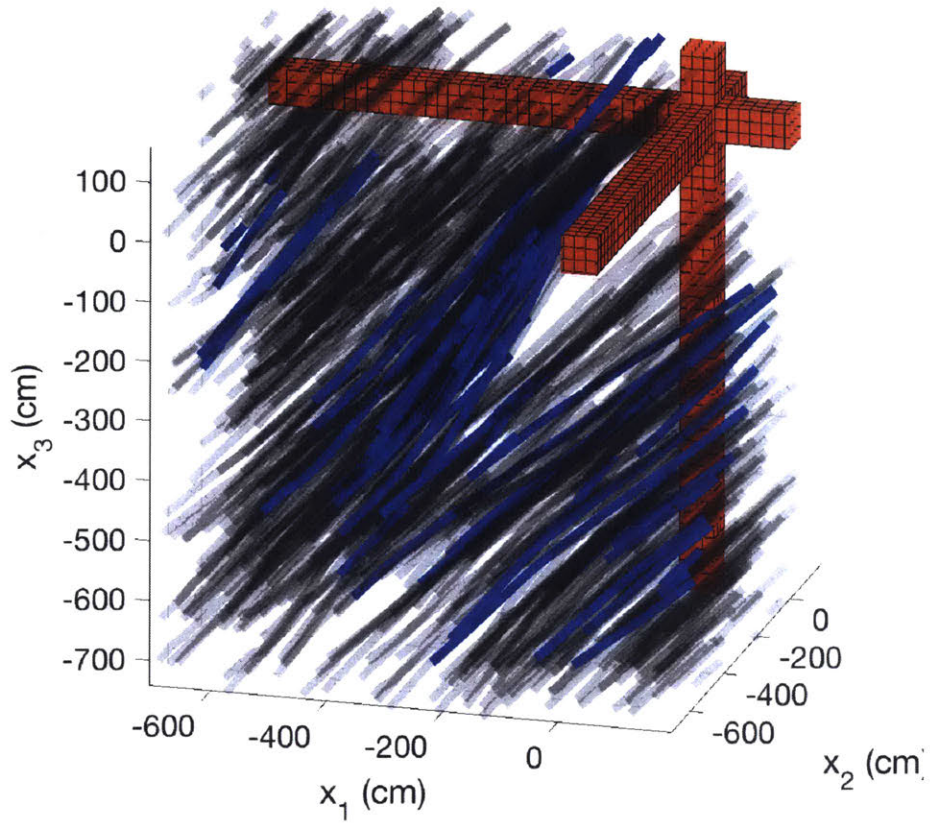
(b)

Figure 4-6: Experimental results of the supervisor. The solid black and dotted red lines represent the position measurement of the controlled and uncontrolled cars, respectively. The same coloring is used as in Figure 4-3.

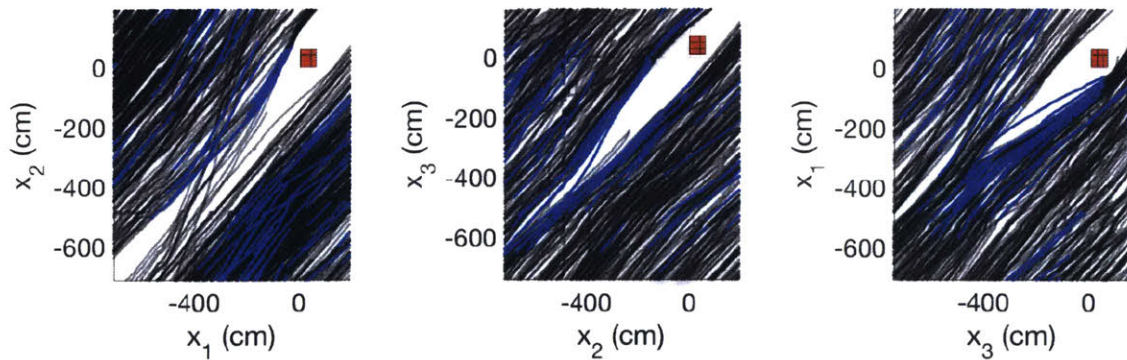
parameters, which are given as follows: $a = (-0.53, -0.30, -0.43) \text{ s}^{-1}$, and $b = (-84.68, -66.43, -49.64) \text{ cm/s}^2$. The bounds of the speed are $v_{\min} = (25, 25, 40) \text{ cm/s}$ and $v_{\max} = (200, 200, 130) \text{ cm/s}$, and those of the motor input u'_i (PWM) are $u_{\min} = (105, 105, 130)$ and $u_{\max} = (170, 165, 150)$. The bounds for disturbance inputs and measurement errors are chosen from their distributions obtained from experiments. The bounds for d_x are $(-5, -3, -3) \text{ cm/s}$ and $(3, 4, 3) \text{ cm/s}$ and for d'_y , $(-4, -2, -3) \text{ cm/s}^2$ and $(2, 3, 2.5) \text{ cm/s}^2$. The measurement error bounds are $\delta_{i,\min} = (-25 \text{ cm}, -25 \text{ cm/s})$ and $\delta_{i,\max} = (25 \text{ cm}, 16 \text{ cm/s})$ for all i . For computer simulations in Section 4.5.1, the parameter values of car 1 and car 3 are used for controlled and uncontrolled vehicles, respectively.

Figure 4-6 depicts two experimental results near the intersection located at $(0, 65) \text{ cm}$ for all cars. In the first scenario (a), panel 1 shows that the uncontrolled car (dotted red circle) approaches the intersection earlier than the controlled cars (solid blue circles). Thus, the supervisor forces the controlled cars to decelerate to avoid a conflict. Panel 2 shows that the conflict is resolved, and one controlled car is crossing the intersection alone without overrides. In the second scenario (b) as shown in panel 1, all three cars are close to the intersection, and the supervisor forces one controlled car to accelerate and the other to decelerate to prevent a collision with the uncontrolled car. Note that the upper state estimate of the last car's position enters the intersection right after the lower state estimate of the uncontrolled car's position has exited, indicating that the override was necessary to avoid the collision. In both scenarios, intersection collisions are averted.

Figure 4-7 depicts 509 trajectories (semitransparent black lines) near the set in which the cars collide (red solid). From the 2D projections in (b), we can confirm that none of the trajectories enters the red solid. We can see that the supervisor overrides cars 1 and 2 when the trajectories would enter the red solid if the trajectories were linear. We observed the failure of 15 out of 509 experiments due to disturbances and measurement errors outside the model bounds. In the 15 experiments, the supervisor terminates before the cars enter the intersection because of incompatibility of the state estimate and measurement. The choice of small bounds for disturbances and



(a)



(b)

Figure 4-7: Position trajectories of cars 1-3. The red region represents a set of positions where cars collide. The trajectories (semitransparent black lines) are in blue when the supervisor intervenes. The 2D projections confirm that no trajectory enters the red region.

measurement errors was necessary in the confined laboratory because otherwise a feasible initial condition may not always exist.

Chapter 5

Supervisor at Multiple Conflict Areas

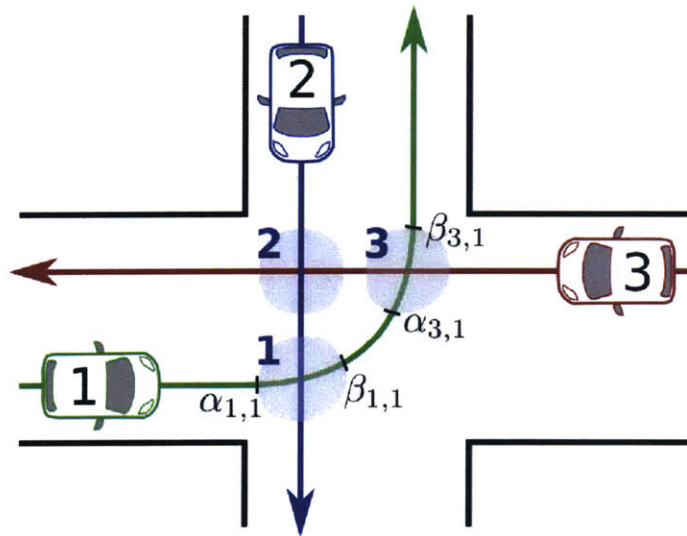


Figure 5-1: An intersection is modeled as a set of conflict areas near which two longitudinal predefined paths intersect. The locations of conflict areas 1 and 3 along the path of vehicle 1 are denoted by $(\alpha_{1,1}, \beta_{1,1})$ and $(\alpha_{3,1}, \beta_{3,1})$, respectively.

Modeling an intersection as a collection of multiple conflict areas enables the design of a less restrictive supervisor than the supervisor in the previous chapter, as mentioned in Chapter 2. For example, in Figure 5-1, the intersection is modeled as a set of conflict areas 1-3. The primary focus of this chapter is to design a supervisor that prevents side collisions among vehicles inside conflict areas. Recall that the bad

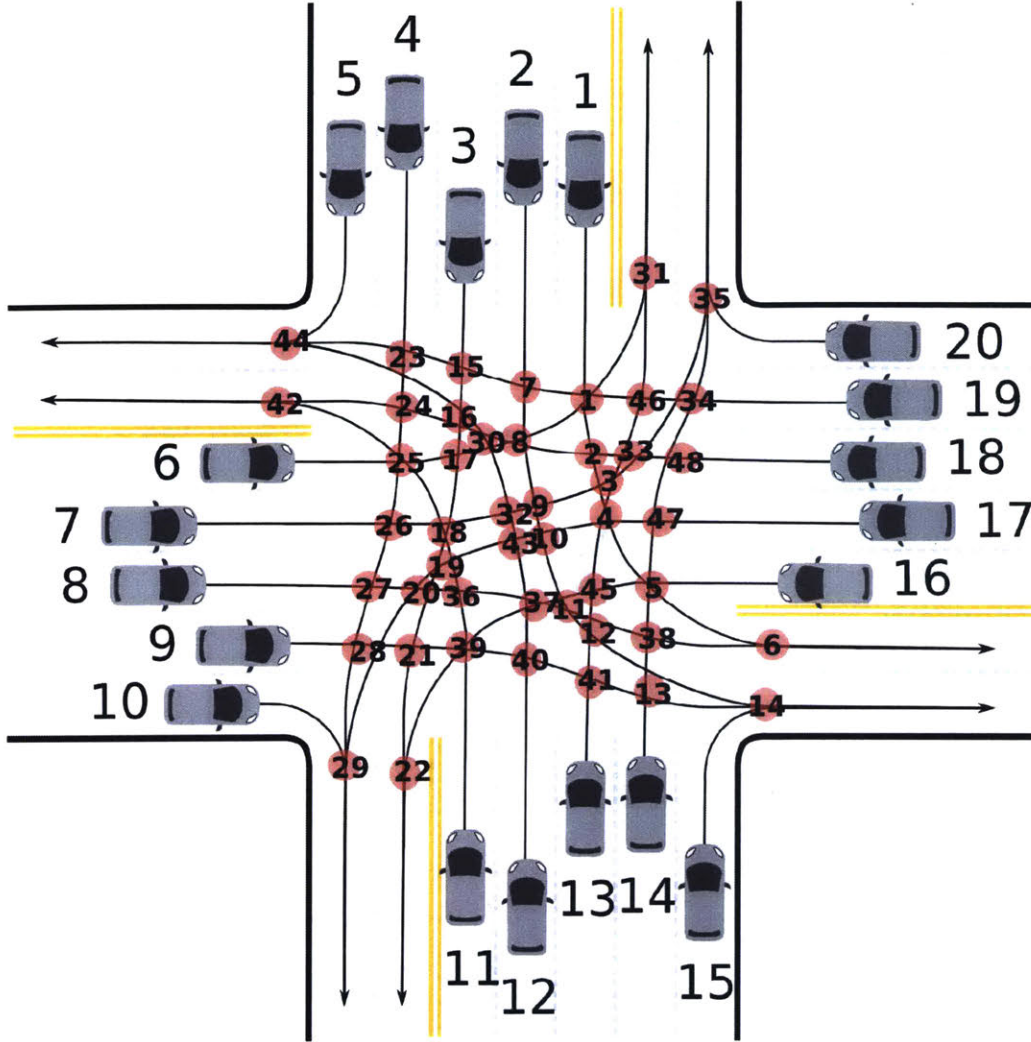


Figure 5-2: General intersection. The safety verification problem in this scenario, which involves 20 vehicles and 48 conflict areas, without considering rear-end collision avoidance on merging paths, can be approximately solved within quantified bounds. This intersection is obtained from [41] to encompass 20 top crash locations in Massachusetts, USA.

set is

$$B := \{\mathbf{x} \in \mathbf{X} : x_j \in (\alpha_{i,j}, \beta_{i,j}), x_{j'} \in (\alpha_{i,j'}, \beta_{i,j'}) \text{ for some } (j, j') \in \mathcal{D}_i\},$$

where $(\alpha_{i,j}, \beta_{i,j}) \subset X_j \subset \mathbb{R}$ is the location of conflict area i on the path of vehicle j , and $(j, j') \in \mathcal{D}_i$ denotes that the paths of vehicles j and j' intersect inside conflict area i .

This chapter is organized as follows. In Section 5.1, we present a jobshop scheduling problem, which is equivalent to the safety verification problem (Problem 2.1). However, the jobshop problem is still computationally difficult to solve due to its nonconvexity. In Section 5.2, we formulate two mixed integer linear programming (MILP) problems that provide over- and under- approximations of the solution to the jobshop scheduling problem with quantified approximation bounds. Based on the approximate solutions, in Section 5.3, we design a supervisor, which, although not least restrictive, runs in real-time for scenarios of reasonable size (such as that in Figure 5-2) and ensures safety. Using the scenarios depicted in Figures 5-1 and 5-2, we validate the supervisor via computer simulations in Section 5.4. All the proofs in this chapter are collected in Appendix B.

©2017 IEEE. Reprinted, with permission, from Heejin Ahn and Domitilla Del Vecchio, Safety Verification and Control for Collision Avoidance at Road Intersections, IEEE Transactions on Automatic Control, July 2017.

5.1 Jobshop Scheduling

Consider n vehicles approach an intersection, which consists of m conflict areas. As we mentioned in Chapter 3, instances in jobshop scheduling can be presented by a graph $(\mathcal{N}, \mathcal{C}, \mathcal{D})$ where \mathcal{N} is the set of operation (i, j) , which is a pair of conflict area i and vehicle j , \mathcal{C} is the set of conjunctive arcs, and \mathcal{D} is the set of disjunctive arcs.

The set of operations is defined in advance as

$$\mathcal{N} := \{(i, j) \in \{1, \dots, m\} \times \{1, \dots, n\} : \text{conflict area } i \text{ is on the route of vehicle } j\},$$

because we assume that the paths of vehicles are known in advance. We also define a set $\mathcal{N}_{\mathbf{x}(0)} \subseteq \mathcal{N}$ that contains operations that have not yet been processed given an

initial position $\mathbf{x}(0)$, that is,

$$\mathcal{N}_{\mathbf{x}(0)} := \{(i, j) \in \mathcal{N} : x_j(0) < \beta_{i,j}\}.$$

We define this set because some operations may not be relevant given that a vehicle may have already exited a conflict area, captured by $\beta_{i,j} \leq x_j(0)$.

A first operation set $\mathcal{F} \subseteq \mathcal{N}_{\mathbf{x}(0)}$ and a last operation set $\mathcal{L} \subseteq \mathcal{N}_{\mathbf{x}(0)}$ are defined as

$$\mathcal{F} := \{(i, j) \in \mathcal{N}_{\mathbf{x}(0)} : i = \arg \min_i (\alpha_{i,j} - x_j(0))\},$$

$$\mathcal{L} := \{(i, j) \in \mathcal{N}_{\mathbf{x}(0)} : i = \arg \max_i (\alpha_{i,j} - x_j(0))\}.$$

That is, an operation (i, j) is a first operation if conflict area i is the first conflict area on the path of vehicle j , and is a last operation if conflict area i is the last such conflict area. The sets \mathcal{F} and \mathcal{L} contain the first and last operations of each vehicle, respectively.

We define sets of conjunctive arcs \mathcal{C} and disjunctive arcs \mathcal{D} , which connect two operations in $\mathcal{N}_{\mathbf{x}(0)}$ as follows:

$$\mathcal{C} := \{(i, j) \rightarrow (i', j) \text{ for } (i, j), (i', j) \in \mathcal{N}_{\mathbf{x}(0)} :$$

vehicle j crosses conflict area i and then conflict area i' \},

$$\mathcal{D} := \{(i, j) \leftrightarrow (i, j') \text{ for } (i, j), (i, j') \in \mathcal{N}_{\mathbf{x}(0)} :$$

vehicles j and j' share the same conflict area i \}.

That is, a conjunctive arc in \mathcal{C} represents a sequence of two operations on the path of each vehicle, and a disjunctive arc in \mathcal{D} represents an undetermined sequence of two operations on the same conflict area. We also denote $(i, j) \leftrightarrow (i, j') \in \mathcal{D}$ by

$$(j, j') \in \mathcal{D}_i \text{ or } (j', j) \in \mathcal{D}_i,$$

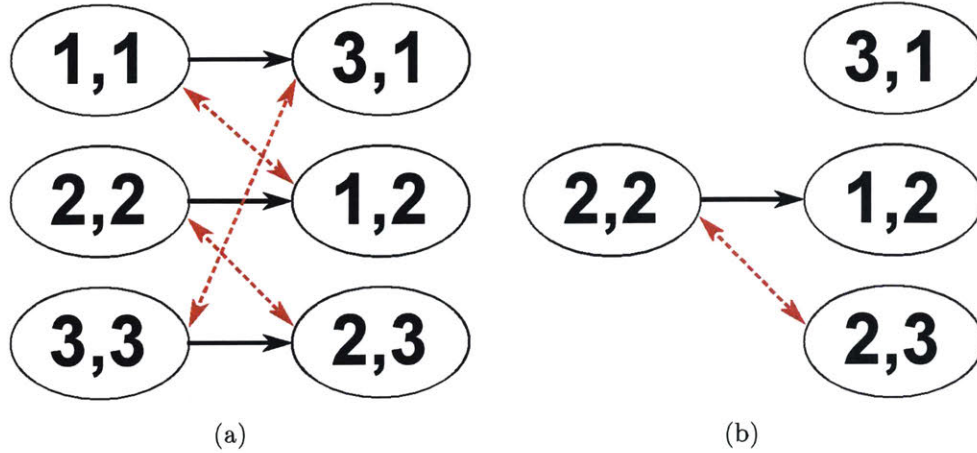


Figure 5-3: (a) All operations of the scenario in Figure 5-1. (b) Operations when $\beta_{1,1} \leq x_1(0) < \beta_{3,1}$, $x_2(0) < \beta_{2,2}$, and $\beta_{3,3} \leq x_3(0) < \beta_{2,3}$. The black solid and red dotted arrows are conjunctive and disjunctive arcs, respectively. See Example 5.1 for more details.

and $(i, j) \rightarrow (i', j) \in \mathcal{C}$ by

$$(i, i') \in \mathcal{C}_j.$$

Example 5.1. In the scenario in Figure 5-1, suppose $\beta_{1,1} \leq x_1(0) < \beta_{3,1}$, $x_2(0) < \beta_{2,2}$, and $\beta_{3,3} \leq x_3(0) < \beta_{2,3}$. The operations are illustrated in Figure 5-3, where the operation sets \mathcal{N} and $\mathcal{N}_{\mathbf{x}(0)}$ are as follows:

$$\mathcal{N} = \{(1, 1), (3, 1), (2, 2), (1, 2), (3, 3), (2, 3)\},$$

$$\mathcal{N}_{\mathbf{x}(0)} = \{(3, 1), (2, 2), (1, 2), (2, 3)\}.$$

Here, the first and last operation sets are

$$\mathcal{F} = \{(3, 1), (2, 2), (2, 3)\}, \quad \mathcal{L} = \{(3, 1), (1, 2), (2, 3)\}.$$

The set of conjunctive arcs is $\mathcal{C} = \{(2, 2) \rightarrow (1, 2)\}$ or denoted by $(2, 1) \in \mathcal{C}_2$, and the set of disjunctive arcs is $\mathcal{D} = \{(2, 2) \leftrightarrow (2, 3)\}$ or denoted by $(2, 3) \in \mathcal{D}_2$.

In this section, we formulate a scheduling problem and present the theorem stating that this problem is equivalent to the safety verification problem (Problem 2.1).

We now introduce the processing characteristics (release times, deadlines, and pro-

cess times) to formulate a jobshop scheduling problem [50]. To simplify the notation, let

$$t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}) := \min_{u_j \in \mathcal{U}_j} \{t : x_j(t, u_j) = \alpha_{i,j}\},$$

and

$$t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}; g(u_j)) := \min_{u_j \in \mathcal{U}_j} \{t : x_j(t, u_j) = \alpha_{i,j} \text{ with constraint } \mathbf{g}(u_j) = \mathbf{0}\}.$$

This is the minimum time to reach $\alpha_{i,j}$ starting from $(x_j(0), \dot{x}_j(0))$ with a given constraint. Similarly, let $t_{\max}(x_j(0), \dot{x}_j(0), \alpha_{i,j}; \mathbf{g}(u_j))$ denote the maximum time to reach $\alpha_{i,j}$ starting from $(x_j(0), \dot{x}_j(0))$ while satisfying constraint $\mathbf{g}(u_j) = \mathbf{0}$.

Let $T_{i,j}$ be a positive number that denotes the time at which vehicle j is scheduled to enter conflict area i , that is, $x_j(T_{i,j}) = \alpha_{i,j}$. Let \mathbf{T} be the set of $T_{i,j}$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$. A jobshop scheduling problem is the problem of determining whether there exists a schedule \mathbf{T} such that an input signal $u_j \in \mathcal{U}_j$ satisfying $x_j(T_{i,j}, u_j) = \alpha_{i,j}$ exists for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ and vehicles never meet inside a conflict area.

Given a schedule \mathbf{T} , the release time $R_{i,j}(\mathbf{T})$ is the soonest time at which vehicle j can enter conflict area i under the constraint that it enters the previous conflict area i' at $T_{i',j}$ (where $(i, i') \in \mathcal{C}_j$). The deadline $D_{i,j}(\mathbf{T})$ is the latest such time. The process time $P_{i,j}(\mathbf{T})$ is the minimum time that vehicle j takes to exit conflict area i under the constraint that it enters the same conflict area at time $T_{i,j}$ and the next conflict area i'' at time $T_{i'',j}$ (where $(i, i'') \in \mathcal{C}_j$). We omit the argument \mathbf{T} if it is clear from context.

Formally, the release time and deadline are defined as follows. Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and a schedule \mathbf{T} , for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$ with $(i', i) \in \mathcal{C}_j$,

$$\begin{aligned} R_{i,j}(\mathbf{T}) &:= t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}; x_j(T_{i',j}, u_j) - \alpha_{i',j}), \\ D_{i,j}(\mathbf{T}) &:= t_{\max}(x_j(0), \dot{x}_j(0), \alpha_{i,j}; x_j(T_{i',j}, u_j) - \alpha_{i',j}). \end{aligned} \tag{5.1}$$

If the constraint is not satisfied, set $R_{i,j} = \infty$ and $D_{i,j} = -\infty$. For all $(i, j) \in \mathcal{F}$, if

$$x_j(0) < \alpha_{i,j},$$

$$\begin{aligned} R_{i,j}(\mathbf{T}) &:= t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}), \\ D_{i,j}(\mathbf{T}) &:= t_{\max}(x_j(0), \dot{x}_j(0), \alpha_{i,j}). \end{aligned} \quad (5.2)$$

If $\alpha_{i,j} \leq x_j(0)$, set $R_{i,j} = D_{i,j} = 0$. Note that for $(i, j) \in \mathcal{F}$, the release time and deadline are independent of \mathbf{T} .

The process time is defined as follows. Given $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and \mathbf{T} , for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{L}$ with $(i, i'') \in \mathcal{C}_j$, if $x_j(0) < \alpha_{i,j}$,

$$P_{i,j}(\mathbf{T}) := t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}; x_j(T_{i,j}, u_j) - \alpha_{i,j}, x_j(T_{i'',j}, u_j) - \alpha_{i'',j}). \quad (5.3)$$

If $\alpha_{i,j} \leq x_j(0)$, set $P_{i,j}(\mathbf{T}) := t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}; x_j(T_{i'',j}, u_j) - \alpha_{i'',j})$. For all $(i, j) \in \mathcal{L}$, if $x_j(0) < \alpha_{i,j}$,

$$P_{i,j}(\mathbf{T}) := t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}; x_j(T_{i,j}, u_j) - \alpha_{i,j}). \quad (5.4)$$

If $\alpha_{i,j} \leq x_j(0)$, set $P_{i,j} := t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j})$. If the constraints are not satisfied, set $P_{i,j} = \infty$.

A jobshop scheduling problem has two decision variables: a schedule \mathbf{T} and a set of binary variables $\mathbf{k} = \{k_{ijj'} \in \{0, 1\}, \forall (i, j) \leftrightarrow (i, j') \in \mathcal{D}\}$ that indicates a sequence of two operations on each disjunctive arc. For example, $k_{ijj'} = 1$ indicates that vehicle j precedes vehicle j' on conflict area i .

Problem 5.1 (jobshop scheduling problem). Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, determine if $s^* = 0$:

$$s^* := \underset{\mathbf{T}, \mathbf{k}}{\text{minimize}} \quad \max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} (T_{i,j} - D_{i,j}(\mathbf{T}), 0)$$

subject to

$$\text{for all } (i, j) \in \mathcal{N}_{\mathbf{x}(0)}, \quad R_{i,j}(\mathbf{T}) \leq T_{i,j}, \quad (\text{P5.1.1})$$

for all $(i, j) \leftrightarrow (i, j') \in \mathcal{D}$,

$$\begin{cases} P_{i,j}(\mathbf{T}) \leq T_{i,j'} + M(1 - k_{ijj'}), \\ P_{i,j'}(\mathbf{T}) \leq T_{i,j} + M(1 - k_{ij'j}), \\ k_{ijj'} + k_{ij'j} = 1. \end{cases} \quad (\text{P5.1.2})$$

where M is a positive and large real number that satisfies

$$\max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} (\beta_{i,j} - x_j(0)) / \dot{x}_{j,\min} \leq M.$$

In the scheduling literature, $\max(T_{i,j} - D_{i,j}(\mathbf{T}), 0)$ is called the *maximum tardiness* [50]. The zero maximum tardiness indicates that a schedule of all operations satisfies the deadline, that is, there is $T_{i,j}$ satisfying $R_{i,j}(\mathbf{T}) \leq T_{i,j} \leq D_{i,j}(\mathbf{T})$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$. This implies the existence of an input signal $u_j \in \mathcal{U}_j$ that makes $x_j(T_{i,j}, u_j) = \alpha_{i,j}$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ by the definitions of $R_{i,j}(\mathbf{T})$ and $D_{i,j}(\mathbf{T})$ and by the assumptions that the space \mathcal{U}_j is connected and that the solution $x_j(t, u_j)$ continuously depends on the input signal u_j (the reasons of these assumptions are found in [16]). Constraint (P5.1.2) says that for two vehicles j and j' that share the same conflict area i (where $(j, j') \in \mathcal{D}_i$), either vehicle j precedes vehicle j' (when $k_{ijj'} = 1$) or the other way around (when $k_{ij'j} = 1$). For example, if $k_{ijj'} = 0$ and $k_{ij'j} = 1$, the first inequality of (P5.1.2) always holds because of the presence of the large constant M satisfying $P_{i,j}(\mathbf{T}) \leq T_{i,j'} + M$ for any \mathbf{T} that satisfies $T_{i,j} \in [R_{i,j}(\mathbf{T}), D_{i,j}(\mathbf{T})]$. The second inequality of (P5.1.2) becomes $P_{i,j'}(\mathbf{T}) \leq T_{i,j}$, implying that after vehicle j' exits conflict area i , vehicle j can enter it. That is, each conflict area is exclusively used by one vehicle at a time. Thus, the existence of such \mathbf{T} and \mathbf{k} that yield $s^* = 0$ is equivalent to the existence of $\mathbf{u} \in \mathcal{U}$ to avoid the bad set. This is the essence of the proof of the following theorem.

Theorem 5.1. *Problem 2.1 is equivalent to Problem 5.1.*

Theorem 5.1 implies the following: given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$,

$$\begin{aligned} s^* = 0 &\implies \text{a safe input signal exists to avoid } B, \\ s^* > 0 &\implies \text{no safe input signal exists to avoid } B. \end{aligned}$$

That is, s^* is the indicator of the vehicles' safety.

While this theorem holds for general dynamics (2.4), Problem 5.1 can be difficult to solve depending on which vehicle dynamics are considered. Problem 5.1 in this paper is a mixed integer nonlinear programming (MINLP) problem, which is notorious for its computational intractability, due to the nonlinear and higher-order vehicle dynamics. To approximately solve Problem 5.1, we formulate two MILP problems that yield lower and upper bounds of s^* , respectively, which can be efficiently solved by a commercially available solver, such as CPLEX [33]. The first MILP problem that computes the lower bound is based on first-order vehicle dynamics, and the second MILP problem that computes the upper bound is based on nonlinear second-order dynamics with a limited input signal space.

5.2 Approximate Verification

In this section, we provide two MILP problems that yield lower and upper bounds of the optimal cost of Problem 5.1, and quantify the approximation errors introduced by the two MILP problems.

5.2.1 Lower Bound Problem

Let us consider first-order vehicle dynamics

$$\dot{y}_j = v_j,$$

where y_j is the position of vehicle j along its path and v_j is the speed input. In this model, we control the speed of vehicles that lies in the space $V_j = [v_{j,\min}, v_{j,\max}]$ with

$\dot{v}_{j,\min} > 0$. The speed signal $v_j : \mathbb{R}_+ \rightarrow V_j$ is piecewise continuous with a countable number of discontinuities. We consider $v_{j,\min} = \dot{x}_{j,\min}$ and $v_{j,\max} = \dot{x}_{j,\max}$ where $\dot{x}_{j,\min}$ and $\dot{x}_{j,\max}$ are the speed bounds for the general dynamics (2.4).

In the lower bound problem, the decision variables are a schedule $\mathbf{t} = \{t_{i,j} \geq 0, (i,j) \in \mathcal{N}_{\mathbf{x}(0)}\}$, a set of process times $\mathbf{p} = \{p_{i,j} \geq 0, (i,j) \in \mathcal{N}_{\mathbf{x}(0)}\}$, and a set of binary variables $\mathbf{k} := \{k_{ijj'} \in \{0,1\}, (j,j') \in \mathcal{D}_i\}$. Note that different from Problem 5.1, the process times are decision variables in the lower bound problem. A schedule and a process time satisfy $y_j(t_{i,j}) = \alpha_{i,j}$ and $y_j(p_{i,j}) = \beta_{i,j}$ for all $(i,j) \in \mathcal{N}_{\mathbf{x}(0)}$.

We define release times $r_{i,j}$ and deadlines $d_{i,j}$ for the lower bound problem, given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and a process time \mathbf{p} , as follows. Let $\mathbf{y}(0) = \mathbf{x}(0)$. For $(i,j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$ with $(i',i) \in \mathcal{C}_j$,

$$r_{i,j}(\mathbf{p}) := p_{i',j} + \frac{\alpha_{i,j} - \beta_{i',j}}{v_{j,\max}}, \quad d_{i,j}(\mathbf{p}) := p_{i',j} + \frac{\alpha_{i,j} - \beta_{i',j}}{v_{j,\min}}. \quad (5.5)$$

For all $(i,j) \in \mathcal{F}$, if $y_j(0) < \alpha_{i,j}$,

$$\begin{aligned} r_{i,j}(\mathbf{p}) &:= t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}), \\ d_{i,j}(\mathbf{p}) &:= t_{\max}(x_j(0), \dot{x}_j(0), \alpha_{i,j}). \end{aligned} \quad (5.6)$$

If $\alpha_{i,j} \leq y_j(0)$, then $r_{i,j} = d_{i,j} = 0$.

Note that for the first operation $(i,j) \in \mathcal{F}$, the release times and deadlines are defined on the general dynamics (2.4). This is because the release times and deadlines even with the general dynamics are linearly dependent (more precisely, independent) on the decision variables \mathbf{p} and thus enable us to write linear constraints. Setting $r_{i,j} = R_{i,j}$ and $d_{i,j} = D_{i,j}$ for the first operation allows a tighter constraint than $r_{i,j} = (\alpha_{i,j} - y_j(0))/v_{j,\max}$ and $d_{i,j} = (\alpha_{i,j} - y_j(0))/v_{j,\min}$.

Problem 5.2 (lower bound problem). Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and $\mathbf{y}(0) = \mathbf{x}(0)$, determine if $s_L^* = 0$:

$$s_L^* := \underset{\mathbf{t}, \mathbf{p}, \mathbf{k}}{\text{minimize}} \quad \max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} (t_{i,j} - d_{i,j}(\mathbf{p}), 0)$$

subject to

$$\text{for all } (i, j) \in \mathcal{N}_{\mathbf{x}(0)}, \quad r_{i,j}(\mathbf{p}) \leq t_{i,j}, \quad (\text{P5.2.1})$$

$$\text{for all } (i, j) \in \mathcal{N}_{\mathbf{x}(0)}, \quad \frac{\beta_{i,j} - \alpha_{i,j}}{v_{j,\max}} \leq p_{i,j} - t_{i,j} \leq \frac{\beta_{i,j} - \alpha_{i,j}}{v_{j,\min}}, \quad (\text{P5.2.2})$$

$$\text{for all } (i, j) \leftrightarrow (i, j') \in \mathcal{D}, \quad \begin{cases} p_{i,j} \leq t_{i,j'} + M(1 - k_{ijj'}), \\ p_{i,j'} \leq t_{i,j} + M(1 - k_{ij'j}), \\ k_{ijj'} + k_{ij'j} = 1. \end{cases} \quad (\text{P5.2.3})$$

where M is a positive and large real number that satisfies

$$\max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} \frac{\beta_{i,j} - y_j(0)}{v_{j,\min}} \leq M.$$

Note that the objective function and constraints (P5.2.1)-(P5.2.3) are linear with the decision variables. Thus, this problem is a mixed integer linear programming (MILP) problem.

The following theorem proves that s_L^* is a lower bound of s^* . The proof of the theorem lies on the fact that the set of trajectories generated by the first-order dynamics is a superset of the trajectories generated by the second-order nonlinear dynamics (2.4).

Theorem 5.2. $s_L^* \leq s^*$

By the above theorem, $s^* > 0$ if $s_L^* > 0$. That is, given a state, if Problem 5.2 does not have a feasible schedule that satisfies (P5.2.1), (P5.2.2), (P5.2.3), and $t_{i,j} \leq d_{i,j}(\mathbf{p})$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$, then Problem 5.1 does not have a schedule that satisfies (P5.1.1), (P5.1.2), and $T_{i,j} \leq D_{i,j}(\mathbf{T})$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$. However, the cost error, $s^* - s_L^*$, cannot be rendered as small as desired due to the simple dynamics (first-order linear) which cannot fully represent the behavior of the actual dynamics (2.4).

5.2.2 Upper Bound Problem

In this section, we relax Problem 5.1 to a MILP problem by considering general dynamics (2.4) on a restricted input signal space. This problem is the problem of determining the existence of a set of times at which vehicles enter their *first conflict area*, assuming that to reach the following conflict areas all vehicles apply maximum input, such that any two vehicles do not meet inside a conflict area. The rationale here is that once a vehicle enters an intersection, the driver tries to exit as soon as possible.

We define $\alpha_{j,\min}$ to denote the first conflict area as follows:

$$\alpha_{j,\min} := \min_{(i,j) \in \mathcal{N}} \alpha_{i,j}.$$

Recall that \mathcal{N} is the set of all operations independent of an initial state.

In the upper bound problem, the time to reach the first conflict area is a decision variable, as opposed to Problems 5.1 and 5.2 whose decision variables are the entering times for all conflict areas. This decision variable, also called a schedule and denoted by $\mathbf{T}^{\mathcal{F}} = \{T_{i,j}^{\mathcal{F}} \geq 0, (i,j) \in \mathcal{F}\}$, satisfies $x_j(T_{i,j}^{\mathcal{F}}) = \alpha_{i,j}$ if $x_j(0) < \alpha_{i,j}$ and otherwise $T_{i,j}^{\mathcal{F}} = 0$. Another decision variable is a set of binary variables $\mathbf{k} = \{k_{ijj'} \in \{0, 1\}, (j, j') \in \mathcal{D}_i\}$ as introduced also in the previous two problems.

The release times and deadlines are defined only for the first operation as follows.

Definition 5.1. Given an initial condition $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, release times $\bar{R}_{i,j}$ and deadlines $\bar{D}_{i,j}$ are defined for $(i,j) \in \mathcal{F}$ as follows. If $x_j(0) < \alpha_{j,\min}$,

$$\begin{aligned} \bar{R}_{i,j} &:= t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{j,\min}), \\ \bar{D}_{i,j} &:= t_{\max}(x_j(0), \dot{x}_j(0), \alpha_{j,\min}). \end{aligned} \tag{5.7}$$

If $\alpha_{j,\min} \leq x_j(0) < \alpha_{i,j}$,

$$\bar{R}_{i,j} := t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}), \quad \bar{D}_{i,j} = \bar{R}_{i,j}. \tag{5.8}$$

If $\alpha_{i,j} \leq x_j(0)$, set $\bar{R}_{i,j} = \bar{D}_{i,j} = 0$.

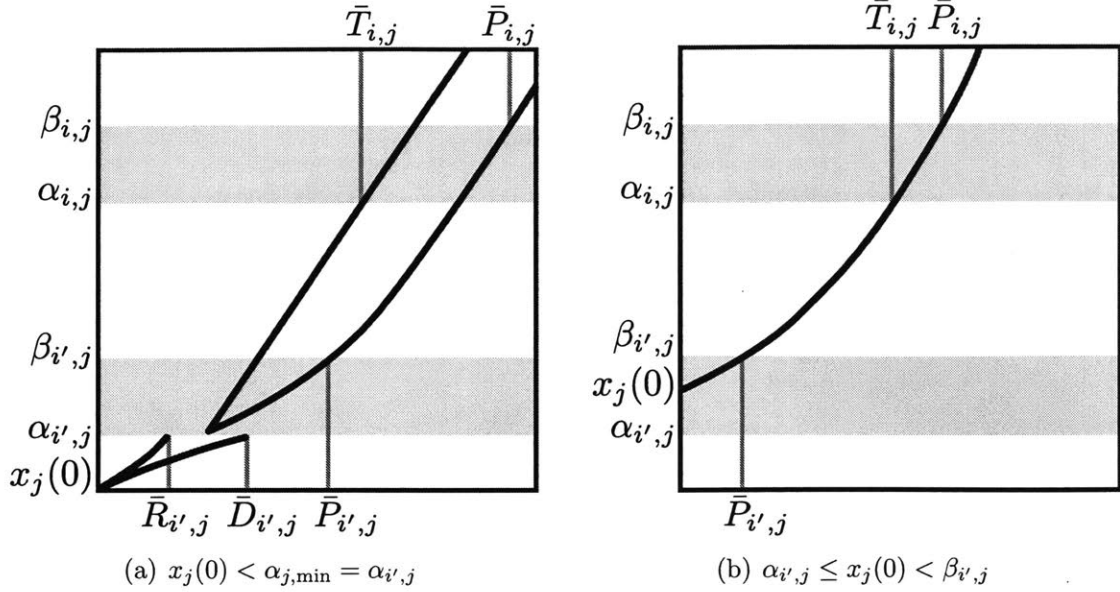


Figure 5-4: Illustration of Definitions 5.1 and 5.2, where the x -axis represents time and the y -axis represents the position. Suppose $(i', j) \in \mathcal{F}$ and $(i', j) \rightarrow (i, j) \in \mathcal{C}$. We can compute $\bar{P}_{i',j}, \bar{T}_{i,j}, \bar{P}_{i,j}$ by considering the maximum input inside the intersection.

Note that $\bar{R}_{i,j} = R_{i,j}$ and $\bar{D}_{i,j} = D_{i,j}$ if $x_j(0) < \alpha_{j,\min}$, and $\bar{R}_{i,j} = \bar{D}_{i,j} = R_{i,j}$ if $x_j(0) \geq \alpha_{j,\min}$. The release time $\bar{R}_{i,j}$ and the deadline $\bar{D}_{i,j}$ depend only on the initial condition $(x_j(0), \dot{x}_j(0))$, not on the decision variable $\mathbf{T}^{\mathcal{F}}$.

Given a schedule $\mathbf{T}^{\mathcal{F}}$, we define $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}})$ for $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ as follows. When vehicle j 's location is before the intersection ($x_j(0) < \alpha_{j,\min}$), $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}})$ represent the minimum times at which vehicle j can enter and exit conflict area i , respectively, *no matter what speed* it has at $T_{i',j}^{\mathcal{F}}$ where (i', j) is the first operation. When vehicle j is in the intersection ($\alpha_{j,\min} \leq x_j(0)$), $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}})$ are the minimum times at which it enters and exits conflict area i , respectively. These are formally defined as follows.

Definition 5.2. Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and a schedule $\mathbf{T}^{\mathcal{F}}$, we define $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}})$ for $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ as follows. If $x_j(0) < \alpha_{j,\min}$, for $(i, j) \in \mathcal{F}$,

$$\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}}) = T_{i,j}^{\mathcal{F}}, \quad \bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}}) = T_{i,j}^{\mathcal{F}} + t_{\min}(\alpha_{i,j}, \dot{x}_{j,\min}, \beta_{i,j}), \quad (5.9)$$

and for $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$ with the first operation (i', j) such that $(i', j) \in \mathcal{F}$ and

$$\alpha_{i',j} = \alpha_{j,\min},$$

$$\begin{aligned}\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i',j}^{\mathcal{F}} + t_{\min}(\alpha_{j,\min}, \dot{x}_{j,\max}, \alpha_{i,j}), \\ \bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i',j}^{\mathcal{F}} + t_{\min}(\alpha_{j,\min}, \dot{x}_{j,\min}, \beta_{i,j}).\end{aligned}\tag{5.10}$$

If $\alpha_{j,\min} \leq x_j(0)$, for $(i, j) \in \mathcal{F}$,

$$\begin{aligned}\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i,j}^{\mathcal{F}}, \\ \bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i,j}^{\mathcal{F}} - \bar{R}_{i,j} + t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}),\end{aligned}\tag{5.11}$$

and for $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$ with the first operation (i', j) such that $(i', j) \in \mathcal{F}$,

$$\begin{aligned}\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i',j}^{\mathcal{F}} - \bar{R}_{i',j} + t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}), \\ \bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}}) &= T_{i',j}^{\mathcal{F}} - \bar{R}_{i',j} + t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}).\end{aligned}\tag{5.12}$$

In (5.11) and (5.12), for $(i', j) \in \mathcal{F}$, $T_{i',j}^{\mathcal{F}} - \bar{R}_{i',j} = 0$ if $T_{i',j}^{\mathcal{F}} \in [\bar{R}_{i',j}, \bar{D}_{i',j}]$ because $\bar{R}_{i',j} = \bar{D}_{i',j}$ by (5.8).

The definitions of $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}})$ are illustrated in Figure 5-4. Note that the concept of applying the maximum input inside the intersection is in the expression t_{\min} in Definition 5.2 because the maximum input leads to the soonest time to reach a position by the monotonicity. Also, note that the parameters in Definitions 5.1 and 5.2 are linear functions with respect to the decision variable $\mathbf{T}^{\mathcal{F}}$ because the terms involving t_{\min} and t_{\max} are independent of the decision variable. We can easily compute the expressions t_{\min} and t_{\max} by finding a time-optimal trajectory, which is obtained by applying extreme control inputs.

Using these definitions, we formulate the upper bound problem.

Problem 5.3 (upper bound problem). Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, determine if $s_U^* = 0$:

$$s_U^* := \underset{\mathbf{T}^{\mathcal{F}}, \mathbf{k}}{\text{minimize}} \max_{(i,j) \in \mathcal{F}} (T_{i,j}^{\mathcal{F}} - \bar{D}_{i,j}, 0)$$

subject to

$$\text{for all } (i, j) \in \mathcal{F}, \quad \bar{R}_{i,j} \leq T_{i,j}^{\mathcal{F}}, \quad (\text{P5.3.1})$$

$$\text{for all } (i, j) \leftrightarrow (i, j') \in \mathcal{D}, \quad \begin{cases} \bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}}) \leq \bar{T}_{i,j'}(\mathbf{T}^{\mathcal{F}}) + M(1 - k_{ijj'}), \\ \bar{P}_{i,j'}(\mathbf{T}^{\mathcal{F}}) \leq \bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}}) + M(1 - k_{ij'j}), \\ k_{ijj'} + k_{ij'j} = 1, \end{cases} \quad (\text{P5.3.2})$$

where M is a positive and large real number that satisfies

$$\max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} (\beta_{i,j} - x_j(0)) / \dot{x}_{j,\min} \leq M.$$

As the constraints are in linear forms with the decision variables $\mathbf{T}^{\mathcal{F}}$ and \mathbf{k} , the problem is a MILP problem.

We show that $s_{\mathcal{U}}^*$ in Problem 5.3 is an upper bound of s^* in Problem 5.1 in the sense that $s^* \leq \bar{M}s_{\mathcal{U}}^*$ for an arbitrarily large positive number \bar{M} . If $s_{\mathcal{U}}^* > 0$, this inequality is trivial, while it is not if $s_{\mathcal{U}}^* = 0$ because then s^* must also be zero. In the following theorem, therefore, we will show that $s_{\mathcal{U}}^* = 0$ implies $s^* = 0$ for any state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$.

Theorem 5.3. $s_{\mathcal{U}}^* = 0 \Rightarrow s^* = 0$.

The proof relies on the fact that the set of trajectories generated by the second-order nonlinear dynamics with a restricted input signal space is a subset of the trajectories generated by the second-order nonlinear dynamics with the full input signal space. Thus, a feasible schedule of Problem 5.3 can be realized by the second-order nonlinear dynamics, thereby guaranteeing the existence of a feasible schedule of Problem 5.1.

By Theorems 5.2 and 5.3, we have

$$s_L^* > 0 \Rightarrow s^* > 0 \quad \text{and} \quad s_{\mathcal{U}}^* = 0 \Rightarrow s^* = 0.$$

That is, from Problems 5.2 and 5.3, which can be solved with a commercial solver

Table 5.1: s^* is bounded below by s_L^* and above by s_U^* .

	s_L^*	s_U^*	s^*
Case I	\cdot	0	0
Case II	0	+	?
Case III	+	\cdot	+

such as CPLEX, we can find the solution to Problem 5.1, as shown in Table 5.1. In Case I, if $s_U^* = 0$, we know that $s^* = 0$. In Case III, if $s_L^* > 0$, we know that $s^* > 0$. However, when $s_L^* = 0$ and $s_U^* > 0$, represented by Case II in the table, s^* is not exactly determined. In this case, we can quantify the approximation bounds.

5.2.3 Approximation Bounds

To determine the solution to Problem 5.1, the solutions to the approximate problems (Problems 5.2 and 5.3) are used when $s_L^* > 0$ or $s_U^* = 0$. However, when $s_L^* = 0$ and $s_U^* > 0$, the approximate problems cannot determine the solution to Problem 5.1. To represent the difference between the approximate and exact solutions in this case, we introduce a *shrunk bad set* and an *inflated bad set*. A shrunk bad set is such that if it were considered in place of the bad set in Problem 5.1, then whenever $s_L^* = 0$ Problem 5.1 would return $s^* = 0$, meaning there exists an input signal with which the state trajectory can avoid the shrunk bad set. An inflated bad set is such that if it were considered in place of the bad set in Problem 5.1, then whenever $s_U^* > 0$ Problem 5.1 would return $s^* > 0$, meaning that for all input signals the state trajectory cannot avoid the inflated bad set. In other words, when $s_L^* = 0$ and $s_U^* > 0$, the shrunk bad set leads to an under-approximation of the solution to Problem 5.1, and the inflated bad set leads to an over-approximation of the solution to Problem 5.1. Thus, the approximation bounds can be quantified in terms of the discrepancy between the shrunk and inflated bad sets.

To construct the shrunk and inflated bad sets, we define a shrunk conflict area and an inflated conflict area. In the definitions, let us consider $\mathcal{F} \subseteq \mathcal{N}$ because the bad sets are geometric objects independent of an initial state of vehicles. See Figure 5-5.

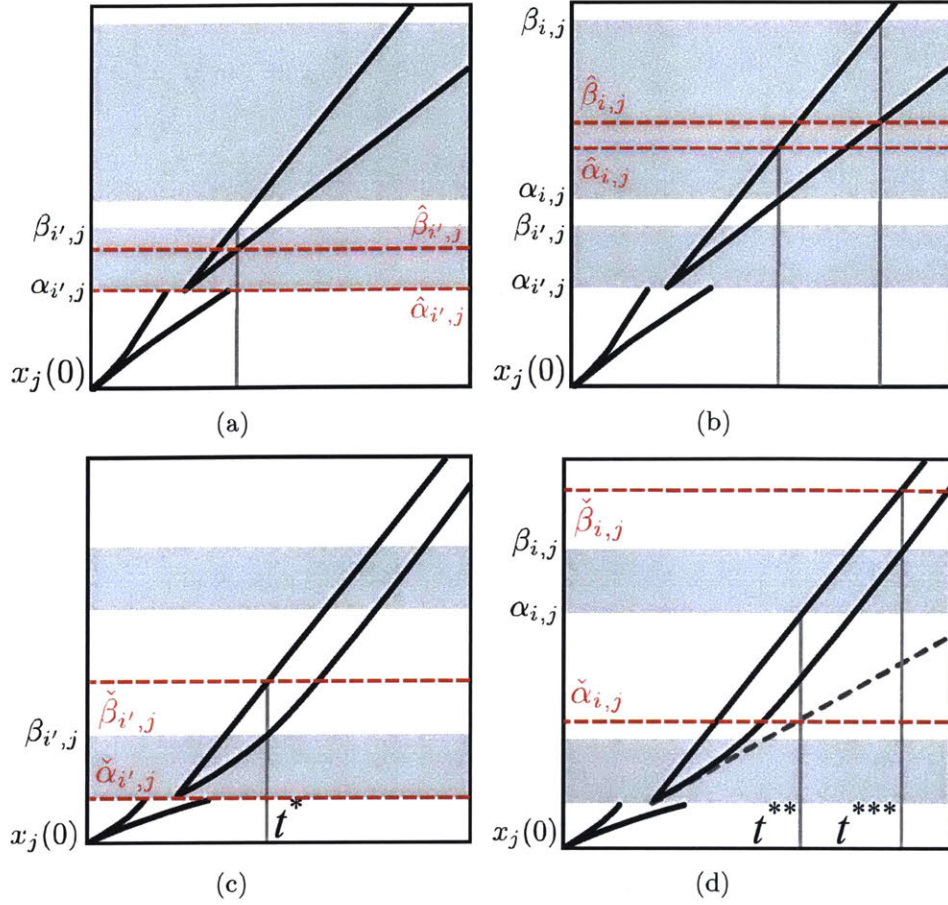


Figure 5-5: Shrunk and Inflated conflict areas for $(i', j) \in \mathcal{F}$ and $(i', i) \in \mathcal{C}_j$. Figures (a)-(d) illustrate (5.13)-(5.16), respectively. By definition, $(\hat{\alpha}_{i',j}, \hat{\beta}_{i',j}) \subseteq (\alpha_{i',j}, \beta_{i',j}) \subseteq (\check{\alpha}_{i',j}, \check{\beta}_{i',j})$ and $(\hat{\alpha}_{i,j}, \hat{\beta}_{i,j}) \subseteq (\alpha_{i,j}, \beta_{i,j}) \subseteq (\check{\alpha}_{i,j}, \check{\beta}_{i,j})$.

Definition 5.3. A shrunken conflict area $(\hat{\alpha}_{i,j}, \hat{\beta}_{i,j})$ for all $(i, j) \in \mathcal{N}$ is defined as follows. For all $(i, j) \in \mathcal{F}$,

$$\begin{aligned} \hat{\alpha}_{i,j} &= \alpha_{i,j}, \\ \hat{\beta}_{i,j} &= \min_{u_j \in \mathcal{U}_j} x_j \left(\frac{\beta_{i,j} - \alpha_{i,j}}{\dot{x}_{j,\max}}, u_j, \alpha_{i,j}, \dot{x}_{j,\min} \right). \end{aligned} \quad (5.13)$$

For all $(i, j) \in \mathcal{N} \setminus \mathcal{F}$ with the first operation $(i', j) \in \mathcal{F}$,

$$\begin{aligned} \hat{\alpha}_{i,j} &= \max_{u_j \in \mathcal{U}_j} x_j \left(\frac{\alpha_{i,j} - \alpha_{i',j}}{\dot{x}_{j,\min}}, u_j, \alpha_{i',j}, \dot{x}_{j,\max} \right), \\ \hat{\beta}_{i,j} &= \min_{u_j \in \mathcal{U}_j} x_j \left(\frac{\beta_{i,j} - \alpha_{i',j}}{\dot{x}_{j,\max}}, u_j, \alpha_{i',j}, \dot{x}_{j,\min} \right). \end{aligned} \quad (5.14)$$

If $\hat{\alpha}_{i,j} \geq \hat{\beta}_{i,j}$, set $(\hat{\alpha}_{i,j}, \hat{\beta}_{i,j}) = \emptyset$.

Definition 5.4. An inflated conflict area $(\check{\alpha}_{i,j}, \check{\beta}_{i,j})$ for all $(i, j) \in \mathcal{N}$ is defined as follows. For all $(i, j) \in \mathcal{F}$,

$$\check{\alpha}_{i,j} = \alpha_{i,j}, \quad \check{\beta}_{i,j} = \max_{u_j \in \mathcal{U}_j} x_j(t^*, u_j, \alpha_{i,j}, \dot{x}_{j,\max}), \quad (5.15)$$

where $t^* = t_{\min}(\alpha_{i,j}, \dot{x}_{j,\min}, \beta_{i,j})$. For all $(i, j) \in \mathcal{N} \setminus \mathcal{F}$ with the first operation $(i', j) \in \mathcal{F}$,

$$\check{\alpha}_{i,j} = \min_{u_j \in \mathcal{U}_j} x_j(t^{**}, u_j, \alpha_{i',j}, \dot{x}_{j,\min}), \quad \check{\beta}_{i,j} = \max_{u_j \in \mathcal{U}_j} x_j(t^{***}, u_j, \alpha_{i',j}, \dot{x}_{j,\max}), \quad (5.16)$$

where $t^{**} = t_{\min}(\alpha_{j,\min}, \dot{x}_{j,\max}, \alpha_{i,j})$ and $t^{***} = t_{\min}(\alpha_{j,\min}, \dot{x}_{j,\min}, \beta_{i,j})$. Note that t^*, t^{**} , and t^{***} are the same as the added times in (5.9) and (5.10).

A shrunken bad set \hat{B} and an inflated bad set \check{B} are defined as follows:

$$\hat{B} := \{\mathbf{x} \in \mathbf{X} : \text{for some } (j, j') \in \mathcal{D}_i, x_j \in (\hat{\alpha}_{i,j}, \hat{\beta}_{i,j}) \text{ and } x_{j'} \in (\hat{\alpha}_{i,j'}, \hat{\beta}_{i,j'})\}. \quad (5.17)$$

$$\check{B} := \{\mathbf{x} \in \mathbf{X} : \text{for some } (j, j') \in \mathcal{D}_i, x_j \in (\check{\alpha}_{i,j}, \check{\beta}_{i,j}) \text{ and } x_{j'} \in (\check{\alpha}_{i,j'}, \check{\beta}_{i,j'})\}. \quad (5.18)$$

It can be checked that

$$\hat{B} \subseteq B \subseteq \check{B}$$

by showing that $(\hat{\alpha}_{i,j}, \hat{\beta}_{i,j}) \subseteq (\alpha_{i,j}, \beta_{i,j}) \subseteq (\check{\alpha}_{i,j}, \check{\beta}_{i,j})$ for all $(i, j) \in \mathcal{N}$.

In the following theorems, we prove that 1) $s_L^* = 0$ implies that there exists an input signal such that the state trajectory can avoid the shrunken bad set \hat{B} , and 2) $s_U^* > 0$ implies that for all input signals, the state trajectory cannot avoid the inflated bad set \check{B} . The proofs of the theorems are collected in Appendix B.

Theorem 5.4. *Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, if $s_L^* = 0$, then there exists an input signal $\mathbf{u} \in \mathcal{U}$ such that $\mathbf{x}(t, \mathbf{u}) \notin \hat{B}$ for all $t \geq 0$.*

Theorem 5.5. *Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, if $s_U^* > 0$, then for all input signals $\mathbf{u} \in \mathcal{U}$, $\mathbf{x}(t, \mathbf{u}) \in \tilde{B}$ for some $t \geq 0$.*

If we consider any strict superset of the shrunken bad set, there exists an initial condition such that Theorem 5.4 does not hold. Similarly, if we consider any strict subset of the inflated bad set, there exists an initial condition such that Theorem 5.5 does not hold. This is shown through the following example.

Example 5.2. Two vehicles with dynamics $\ddot{x}_j = u_j$, $u_j \in [-2, 2]$ m/s² are approaching a single conflict area located at (5, 7) m along the path of each vehicle. A given state is $x_1(0) = 0$ m, $x_2(0) = -0.99$ m, and $\dot{x}_1(0) = \dot{x}_2(0) = 5$ m/s where $\dot{x}_1, \dot{x}_2 \in [1, 5]$ m/s. Then, s_U^* in Problem 5.3 is positive because the optimal solution is $(T_{1,1}^{\mathcal{F}}, \bar{P}_{1,1}) = (1, 2)$ s and $(T_{1,2}^{\mathcal{F}}, \bar{P}_{1,2}) = (2, 3)$ s, but $\bar{D}_{1,2} = 1.99$ s. The inflated conflict area is (5, 10) m by (5.15).

There is no input signal such that the system's state avoids the inflated bad set because at $t = 1$ s, vehicle 1 enters the bad set with the maximum speed and at $t = 2$ s, vehicle 1 can exit the inflated bad set. At $t = 2$ s, vehicle 2 cannot enter the bad set because $D_{1,2} = \bar{D}_{1,2} = 1.99$ s. However, if a strict subset of the inflated conflict area, say (5, 9.94) m, is considered, vehicle 1 exits the subset at $t = 1.988$ s and thus vehicle 2 can enter the bad set no later than its deadline. This means that an input signal exists to avoid the subset of the inflated conflict area.

5.3 Supervisor Algorithm

Based on the results of Section 5.2, we can design a supervisor that is activated when a future collision is detected inside the inflated conflict areas. The structure of the supervisor is illustrated in Figure 5-6.

Let APPROXVERIFICATION($\mathbf{x}(0), \dot{\mathbf{x}}(0)$) be an algorithm solving Problem 5.3 given an initial state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$. Let APPROXVERIFICATION return $(s_U^*, \mathbf{T}^{\mathcal{F}*})$ where $\mathbf{T}^{\mathcal{F}*}$ is the optimal solution.

The supervisory algorithm runs in discrete time with a time step τ . At time $k\tau$, it receives the measurement of the state $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ and the desired input $\mathbf{u}_d^k \in \mathbf{U}$,

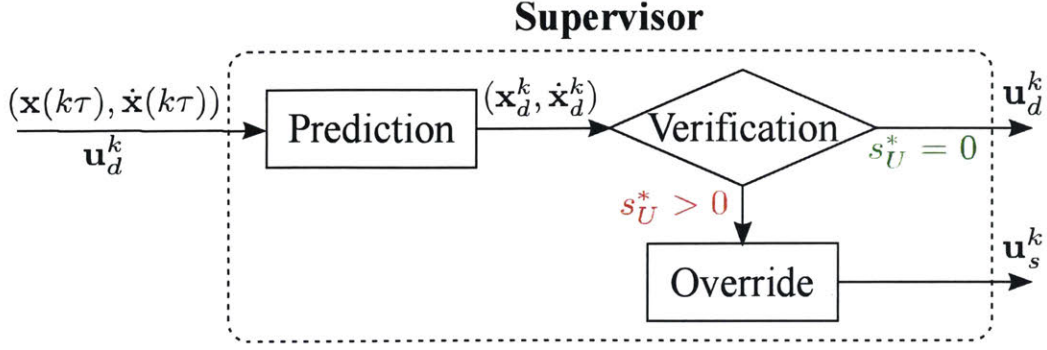


Figure 5-6: The supervisor overrides the vehicles if Problem 5.3 returns $s_U^* > 0$, given the desired state $(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k)$.

which is a vector of inputs that the drivers are applying at time $k\tau$. We also denote by \mathbf{u}_d^k a map $t \in [0, \tau) \mapsto \mathbf{u}_d^k$ if there is no confusion. The supervisor is implemented by the following algorithm.

Algorithm 5-7 Supervisory control algorithm at $t = k\tau$

- 1: **procedure** SUPERVISOR($\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k$)
 - 2: $\mathbf{x}_d^k \leftarrow \mathbf{x}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
 - 3: $\dot{\mathbf{x}}_d^k \leftarrow \dot{\mathbf{x}}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
 - 4: $(s_U^*, \mathbf{T}_1^{\mathcal{F}^*}) = \text{APPROXVERIFICATION}(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k)$
 - 5: **if** $s_U^* = 0$ **then**
 - 6: $\mathbf{u}_s^{k+1} \leftarrow \sigma(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k, \mathbf{T}_1^{\mathcal{F}^*})$
 - 7: **return** \mathbf{u}_d^k
 - 8: **else**
 - 9: $\mathbf{x}_s^k \leftarrow \mathbf{x}(\tau, \mathbf{u}_s^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
 - 10: $\dot{\mathbf{x}}_s^k \leftarrow \dot{\mathbf{x}}(\tau, \mathbf{u}_s^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
 - 11: $(\cdot, \mathbf{T}_2^{\mathcal{F}^*}) = \text{APPROXVERIFICATION}(\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k)$
 - 12: $\mathbf{u}_s^{k+1} \leftarrow \sigma(\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k, \mathbf{T}_2^{\mathcal{F}^*})$
 - 13: **return** \mathbf{u}_s^k restricted to time $[0, \tau)$
-

The measurement $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ is used to predict the desired state at the next time step, which is denoted by $(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k)$ (lines 2 and 3). We solve Problem 5.3 to see if the system's state, starting from the desired state, can avoid the bad set (line 4).

If APPROXVERIFICATION($\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k$) returns $\mathbf{T}_1^{\mathcal{F}^*}$ that makes $s_U^* = 0$ (line 5), we can find a safe input signal by defining an input generator function $\sigma : \mathbf{X} \times \dot{\mathbf{X}} \times \mathbb{R}^n \rightarrow \mathcal{U}$ as follows:

$$\sigma(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k, \mathbf{T}^{\mathcal{F}^*}) \in \{\mathbf{u} \in \mathcal{U} : \text{for all } (i, j) \in \mathcal{F}, \\ x_j(T_{i,j}^{\mathcal{F}^*}, u_j, x_{d,j}^k, \dot{x}_{d,j}^k) = \alpha_{i,j} \text{ and } u_j(t) = u_{j,\max} \forall t \geq T_{i,j}^{\mathcal{F}^*}\},$$

where $x_{d,j}^k$ and $\dot{x}_{d,j}^k$ denote the j -th entries of \mathbf{x}_d^k and $\dot{\mathbf{x}}_d^k$, respectively. The supervisor stores this safe input signal for potential use at the next time step (line 6). Since there is an input signal such that the system avoids entering the bad set from $(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k)$, the supervisor allows the desired input (line 7).

If APPROXVERIFICATION($\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k$) returns $s_U^* > 0$ (line 8), the supervisor overrides the drivers with the safe input signal \mathbf{u}_s^k stored at the previous step. This safe input signal is used to predict the safe state at the next time step, which is denoted by $(\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k)$ (lines 9 and 10). This state is used to generate a safe input signal for potential use at the next time step (lines 11 and 12). We will prove in the next theorem that APPROXVERIFICATION($\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k$) always returns $s_U^* = 0$ and thus a safe input signal is defined by the input generator function σ . The supervisor neglects the desired input \mathbf{u}_d^k and returns the safe input signal \mathbf{u}_s^k that is restricted on time $[0, \tau)$.

The computational complexity of Algorithm 5-7 is determined by the complexity of APPROXVERIFICATION (solving Problem 5.3) in lines 4 and 11. Since Problem 5.3 involves $O(mn(n-1))$ binary variables and n continuous variables, where n is the number of vehicles and m is the number of conflict area, APPROXVERIFICATION has $O(n2^{mn(n-1)})$ asymptotic computational complexity. Although the complexity of APPROXVERIFICATION is combinatorial with the number of vehicles and the number of conflict areas, several algorithmic approaches are available to solve MILP problems [25].

By Theorem 5.5, $s_U^* > 0$ indicates that no input signal exists to avoid the inflated bad set. However, there may exist an input signal to avoid the bad set. Thus, this supervisor is not least restrictive, but is close to the least restrictive one in a quantified

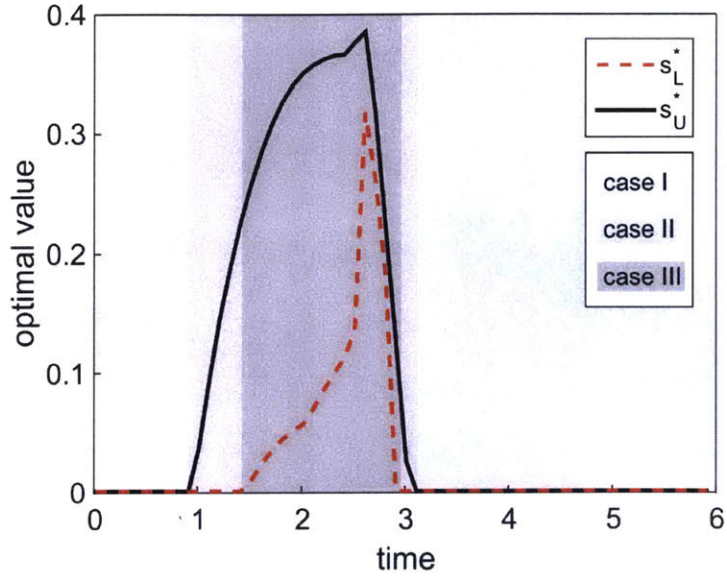


Figure 5-7: Simulation results without the supervisor (Algorithm 5-7) for the scenario in Figure 5-1. Cases I, II, and III denote the same cases in Table 5.1.

bound in the sense that when it overrides vehicles, there is no input signal to avoid the inflated bad set (Theorem 5.5), and when it does not override vehicles, there is an input signal to avoid the bad set (Theorem 5.3).

Theorem 5.6. *Algorithm 5-7 guarantees that the system's state never enters the bad set, that is, $\mathbf{x}(t) \notin B$ for all $t \geq 0$, and is nonblocking.*

5.4 Validation of the Supervisor: Simulations

We implemented Algorithm 5-7 on the cases illustrated in Figures 5-1 and 5-2 to validate its collision avoidance performance and its nonblocking property. We implemented the algorithm on MATLAB and performed simulations on a personal computer consisting of an Intel Core i7 processor at 3.10 GHz and 8 GB RAM.

In the simulations, we consider the vehicle dynamics with a quadratic air drag term

$$\ddot{x}_j = c_1 u_j + c_2 \dot{x}_j^2 + c_3.$$

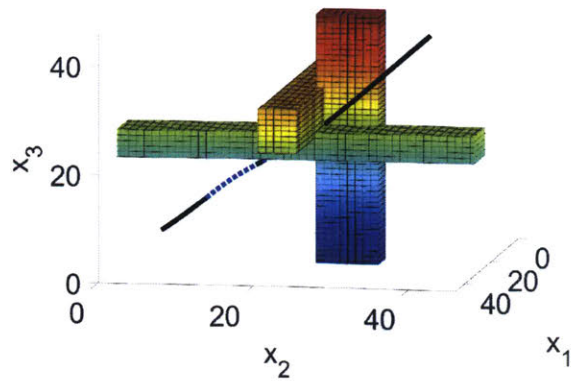
The following parameters are used: $\tau = 0.1, c_1 = 1, c_2 = 0.005, c_3 = 0$. For all

$j \in \{1, \dots, n\}$, $u_{j,\min} = -2, u_{j,\max} = 2, \alpha_{j,\min} = 20$. For all $(i, j) \in \mathcal{N}$, $\beta_{i,j} - \alpha_{i,j} = 5$ and for all $(i, i') \in \mathcal{C}_j$, $\alpha_{i',j} - \alpha_{i,j} = 6$.

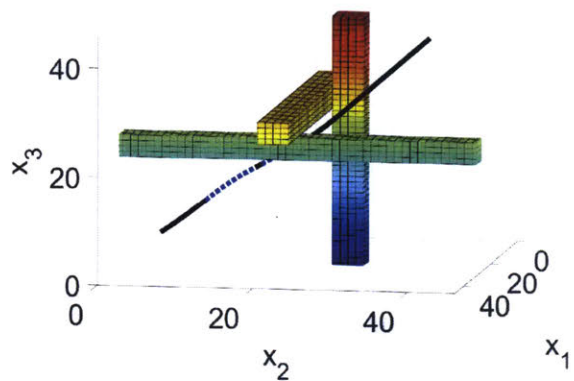
Consider the scenario illustrated in Figure 5-1 with the following initial state and parameters: $\mathbf{x}(0) = (0, 0, 0), \dot{\mathbf{x}}(0) = (10, 8, 8)$, and $\dot{x}_{j,\min} = 8, \dot{x}_{j,\max} = 10$ for all $j \in \{1, \dots, n\}$. Without implementing the supervisor (Algorithm 5-7), we let the vehicles travel with the desired input $\mathbf{u}_d^k = (-2, -2, 2)$ for all k and plot the optimal values of Problems 5.2 and 5.3, shown in Figure 5-7. As proved in Theorems 5.2 and 5.3, $s_U^* = 0$ implies $s_L^* = 0$. With implementing the supervisor, we plot the position trajectory in Figure 5-8(a)-(c). The trajectory (black line) is controlled by the supervisor when $s_U^* > 0$ (dotted line) so that it avoids the bad set (solid in (b)) by Theorem 5.6. Note that the trajectory penetrates the inflated bad set (solid in (a)). This is because although $s_U^* > 0$ implies that there is no input signal to avoid the inflated bad set by Theorem 5.5, $s_U^* = 0$ does not imply that there is an input signal to avoid the inflated bad set, but implies that there is an input signal to avoid the bad set by Theorem 5.3.

Now, let us consider the scenario illustrated in Figure 5-2 with the following initial state and parameters: $\mathbf{x}(0) = (0, -2, 5, -5, 0, 5, 0, 1, 5, 4, 0, -2, 5, 5, 0, 5, -2, 0, -2, 0)$ and $\dot{x}_j(0) = 5, \dot{x}_{j,\min} = 1, \dot{x}_{j,\max} = 10$ for all $j \in \{1, \dots, n\}$, with the desired input $\mathbf{u}_d^k = \mathbf{u}_{\max}$ for all k . The result is shown in Figure 5-9. The trajectory of vehicle 1 (black line) and the trajectories of other vehicles that share the same conflict area (red dotted lines) never stay inside the conflict area simultaneously. The supervisor overrides vehicles when $s_U^* > 0$ (when blue boxes at the bottom appear) to make them cross the intersection without collisions. In the simulations, to determine whether $s_U^* = 0$ or $s_U^* > 0$, we solve the following problem instead of solving Problem 5.3 because solving the following problem (with no cost function) takes less computation time than solving Problem 5.3 (with a cost function) [33]: given an initial state, determine if there exists a feasible solution $(\mathbf{T}^{\mathcal{F}}, \mathbf{k})$ that satisfies (P5.3.1), (P5.3.2), and $T_{i,j}^{\mathcal{F}} \leq \bar{D}_{i,j}$ for all $(i, j) \in \mathcal{F}$. Note that such a feasible solution exists if and only if $s_U^* = 0$. Based on the solution of this feasibility problem, Algorithm 5-7 takes no more than 0.05s per iteration, even in this scenario of realistic size involving 20

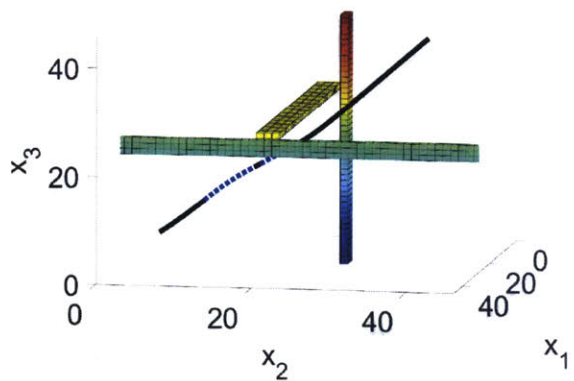
vehicles, 48 conflict areas, and 120 operations on a representative geometry of dangerous intersections. Given that the allocated time step for intelligent transportation systems is 0.1 s [60], this algorithm can be implemented in real-time.



(a) Inflated bad set \hat{B}



(b) Bad set B



(c) Shrunken bad set \hat{B}

Figure 5-8: Simulation results with the supervisor for the scenario in Figure 5-1. The black line represents the trajectory and is the same on each figure. The line turns to the dotted line when the supervisor intervenes to prevent a predicted collision. The solid is (a) the inflated bad set, (b) the bad set, and (c) the shrunken bad set. The supervisor manages the system to avoid entering the bad set.

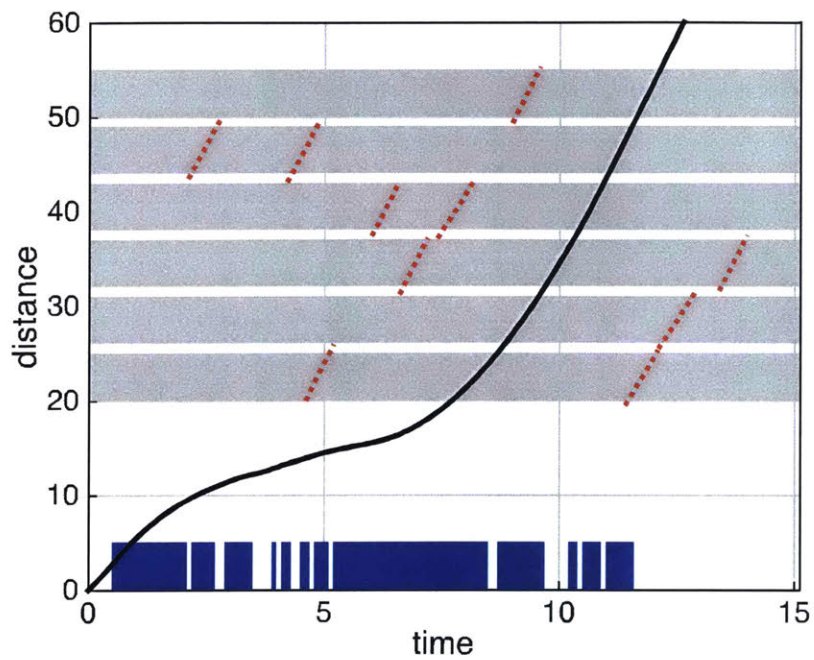


Figure 5-9: Trajectory of vehicle 1 in the scenario of Figure 5-2, which involves 20 vehicles, 48 conflict areas, and 120 operations. The blue boxes represent the times at which the supervisor overrides the vehicles. The red dotted lines are the trajectories of the other vehicles that share the same conflict area. This graph shows that each conflict area is used by only one vehicle at a time.

Chapter 6

Supervisor at Multiple Conflict Areas and Merging/Splitting Paths

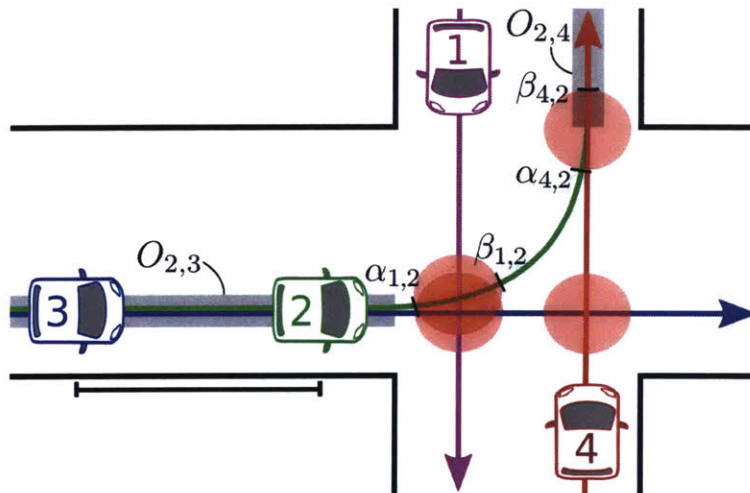


Figure 6-1: Scenario with four vehicles. The paths of vehicles intersect inside four side conflict areas and partly overlap.

The scenarios considered in the previous chapters assume that there is only one vehicle per lane, and thus focus only on side collision avoidance. This chapter presents a novel approach to the design of a supervisor that ensures simultaneous avoidance of side and rear-end collisions. Our approach is to adopt a hierarchical structure for safety verification, as illustrated in Figure 6-2. In this structure, we first formulate a verification problem given the first-order dynamics (abstract system), which

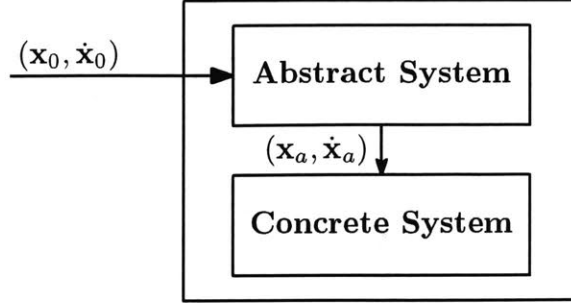


Figure 6-2: Hierarchical structure for the safety verification problem. The verification on an abstract system is used for the verification on a concrete system.

is therefore similar to Problem 5.2 except that the new problem has an additional constraint that limits the instantaneous speed changes of the first-order dynamics so as to represent the behaviors of the second-order dynamics. Then, we use the result to approximately solve the safety verification problem for the actual second-order dynamics (concrete system). In particular, we consider as the concrete system vehicle dynamics that are affine in the input, but otherwise nonlinear:

$$\ddot{x}_j = f(x_j, \dot{x}_j) + b(x_j, \dot{x}_j) u_j, \quad u_j \in U_j := [u_{j,\min}, u_{j,\max}], \quad (6.1)$$

where $f(x_j, \dot{x}_j)$ and $b(x_j, \dot{x}_j)$ are nonlinear functions of the state. We assume that (6.1) has a unique solution that continuously depends on the input signal, and that $-f(x_j, \dot{x}_j)/b(x_j, \dot{x}_j) \in (u_{j,\min}, u_{j,\max})$ for all $x_j \in X_j, \dot{x}_j \in \dot{X}_j$ to ensure that any constant speed in \dot{X}_j is attainable. An example of such dynamics is the longitudinal dynamics given in (2.5).

In this chapter, we present the design of a supervisor that prevents side collisions and rear-end collisions at a general, realistic intersection. We assume that the paths of vehicles approaching an intersection are known in advance. Some of the paths intersect at several points, thereby forming side conflict areas, and overlap, thereby forming splitting or merging regions. Recall that in this scenario, the bad set is

$$B = B_{\text{side}} \cup B_{\text{rear-end}},$$

where

$$B_{\text{side}} := \{\mathbf{x} \in \mathbf{X} : x_j \in (\alpha_{i,j}, \beta_{i,j}), x_{j'} \in (\alpha_{i,j'}, \beta_{i,j'}) \text{ for some } (j, j') \in \mathcal{D}_i\},$$

which is the set of points corresponding to side collisions, and

$$B_{\text{rear-end}} := \{\mathbf{x} \in \mathbf{X} : \text{for some } (j, j') \in \mathcal{O} \text{ if } x_j \in O_{j,j'} \text{ and } x_{j'} \in O_{j',j}, \\ |(x_j - o_{j,j'}) - (x_{j'} - o_{j',j})| < d\},$$

which is the set of points corresponding to rear-end collisions. Here, $(j, j') \in \mathcal{D}_i$ indicates that the paths of vehicles j and j' intersect inside side conflict area i (recall that \mathcal{D} is defined as the set of disjunctive arcs in the graph representation in Section 5.1). The set \mathcal{O} consists of a pair (j, j') if the paths of vehicles j and j' overlap; the rear-end conflict area on the path of vehicle j is denoted by a closed interval $O_{j,j'} \subset X_j$, and that on the path of vehicle j' by a closed interval $O_{j',j} \subset X_{j'}$. Also, $o_{j,j'} = \inf O_{j,j'}$. The notations are represented in Figure 6-1. In the figure, the four red circles indicate side conflict areas, where $(1, 2) \in \mathcal{D}_1$, $(1, 3) \in \mathcal{D}_2$, $(3, 4) \in \mathcal{D}_3$, and $(2, 4) \in \mathcal{D}_4$. The side conflict areas 1 and 4 are located at $(\alpha_{1,2}, \beta_{1,2})$ and $(\alpha_{4,2}, \beta_{4,2})$, respectively, on the path of vehicle 2. The thick lines indicate rear-end conflict areas on the path of vehicle 2, $O_{2,3}$ and $O_{2,4}$.

In order to formulate a verification problem for the abstract system, we adopt an approach based on discretizing the paths of vehicles. Before formulating the discretization-based verification problem in Section 6.2.1, we explain in Section 6.1 why our formulation is computationally more efficient than other formulations based on different discretization schemes. In Section 6.2.2, we use the results of the abstract system's verification problem to solve the verification problem for the concrete system (Problem 2.1) by employing a sliding mode control. Based on the solution, we design a supervisor in Section 6.3 and validate it through computer simulations in Section 6.4.

6.1 Comparison of Discretization Schemes

In this section, we overview the safety verification problem discretized through different discretization schemes to compare the computational complexity of the resulting problem formulation.

In the formulation of the optimization problem, we denote decision variables by $\mathbf{s} = (\mathbf{s}[0], \mathbf{s}[1], \mathbf{s}[2], \dots, \mathbf{s}[N])$, where $\mathbf{s}[k]$ is the variable at (space or time) step k . At step k , let $\mathbf{s}[k] = (s_1[k], s_2[k], \dots, s_n[k])$ where $s_j[k]$ is the variable associated with vehicle j . The positive integer N indicates a finite horizon.

The discretized safety verification problem is written as

$$\begin{aligned} \min \quad & f(\mathbf{s}) \\ \text{subject to} \quad & F_{\text{dynamics}}(\mathbf{s}) \leq \mathbf{0}, \quad F_{\text{rear-end}}(\mathbf{s}) \leq \mathbf{0}, \quad F_{\text{side}}(\mathbf{s}) \leq \mathbf{0}, \end{aligned}$$

with some cost function $f(\mathbf{s}) \in \mathbb{R}$. The constraint $F_{\text{dynamics}} \leq \mathbf{0}$ imposes the dynamical equation, and the constraints $F_{\text{rear-end}} \leq \mathbf{0}$ and $F_{\text{side}} \leq \mathbf{0}$ ensure the avoidance of rear-end and side collisions, respectively. In the following sections, we specify the inequalities for different discretization schemes.

6.1.1 Time Discretization

In uniform time discretization, the decision variable $s_j[k]$ is the dynamical state $(x_j[k], \dot{x}_j[k])$. Vehicle dynamics (2.4) can be discretized in time and written in terms of the state $(x_j[k], \dot{x}_j[k])$ and the input $u_j[k]$ ($F_{\text{dynamics}} \leq \mathbf{0}$).

For rear-end collision avoidance ($F_{\text{rear-end}} \leq \mathbf{0}$) for two different vehicles j and j' , where $(j, j') \in \mathcal{O}$, the constraint is encoded by

$$\begin{aligned} \text{if vehicle } j \text{ precedes vehicle } j', \text{ if } x_j[k] \in O_{j,j'}, \text{ and if } x_{j'}[k] \in O_{j',j}, \\ g(s_j[k], s_{j'}[k]) \geq d. \end{aligned} \tag{6.2}$$

Here, $g(s_j[k], s_{j'}[k]) \geq d$ indicates a set of inequalities imposing that the distance between two vehicles is greater than or equal to the minimum safe distance d . For

side collision avoidance ($F_{\text{side}} \leq 0$) between vehicles j and j' where $(j, j') \in \mathcal{D}_i$, the constraint is encoded by

$$\begin{aligned} &\text{if vehicle } j \text{ precedes vehicle } j' \text{ at side conflict area } i, \text{ if } x_j[k] \in (\alpha_{i,j}, \beta_{i,j}), \\ &\text{then } x_{j'}[k+1] \notin (\alpha_{i,j'}, \beta_{i,j'}). \end{aligned} \quad (6.3)$$

In other words, if vehicle j is inside the side conflict area at time step k , then vehicle j' must not be inside the side conflict area at the next time step.

Note that encoding the “If” statements in (6.2) and (6.3) require additional binary variables because the value of the decision variable \mathbf{s} determines the truth of the statements. To encode “vehicle j precedes vehicle j' ” in side conflict area i , we introduce a binary variable $k_{ijj'}$ such that it is 1 if vehicle j precedes vehicle j' , and 0 otherwise. There are at most $mn(n-1)$ such binary variables, where n is the number of vehicles and m is the number of side conflict areas. To encode $x_j[k] \in O_{j,j'}$ or $x_j[k] \in (\alpha_{i,j}, \beta_{i,j})$, we introduce binary variables to indicate whether $x_j[k]$ is inside a given set. For example, we impose $\gamma_j[k] = 1$ if $x_j[k] \in O_{j,j'}$, and $\gamma_j[k] = 0$ if $x_j[k] \in O_{j,j'}$. Since a binary variable is associated with a vehicle at each step for a rear-end conflict area or a side conflict area, the constraint requires $O(Nnm)$ binary variables where N is the number of time steps.

Solving the problem formulated above tends to require large computation time due to a large number of binary variables involved. In particular, the number of binary variables is nearly proportional to the length of a finite (time) horizon N . A finite horizon cannot be made arbitrarily short to reduce computational loads because the solution (the state trajectory that avoids collisions) over the horizon must guarantee the existence of solutions for all subsequent time steps. One straightforward choice of a horizon is the one in which a vehicle can cross an intersection even in the worst case, such as for each j

$$N_j = \max_{\dot{x}_j \in \dot{X}_j} \left\lceil \frac{x_{j,\max} - x_{0,j}}{\dot{x}_j} \frac{1}{\Delta t} \right\rceil,$$

given an initial position $\mathbf{x}_0 = (x_{0,1}, \dots, x_{0,n})$ and a step size Δt . Since the minimum speed $\dot{x}_{j,\min}$ is close to zero, this horizon is usually very long, and thus, solving the problem requires long computation time. The work in [6] adopted this approach and introduced a shorter finite horizon with a guarantee of future safety, but a shorter finite horizon is applicable only for a simple intersection model (single side conflict area) and for a specific intersection geometry where vehicles can fully stop before an intersection no matter what speed vehicles initially have.

6.1.2 Space Discretization

As opposed to the formulation based on time discretization, uniform space discretization with space step size Δx_j enables a finite (space) horizon without assuming any worst-case behavior of vehicles, which is for each vehicle j

$$N_j = \left\lceil \frac{x_{j,\max} - x_{0,j}}{\Delta x_j} \right\rceil.$$

Furthermore, the total number of binary variables required to write the constraints $F_{\text{rear-end}} \leq \mathbf{0}$ and $F_{\text{side}} \leq \mathbf{0}$ based on space discretization is independent of a finite horizon. This is because the position at step k is $x_{0,j} + k\Delta x_j$, which is fixed given the initial position $x_{0,j}$ and spatial step size Δx_j , and thus, there is no need to introduce any binary variable to indicate whether the position at step k is inside rear-end or side conflict areas. The only required binary variables are the ones indicating the order of vehicles at side conflict areas, whose total number is at most $mn(n-1)$ as discussed in the previous section. Since the asymptotic computational complexity of mixed integer programming problems exponentially increases with the number of discrete variables, the problem in space discretization has more favorable computational complexity than that in time discretization. However, rewriting vehicle dynamics with respect to space introduces nonlinearity in the constraint $F_{\text{dynamics}} \leq \mathbf{0}$, thereby requiring a strong simplifying assumption, such as restricting the bounds of inputs [45], to linearize the constraint.

Here, we do not present explicitly the expressions of the constraints $F_{\text{rear-end}} \leq \mathbf{0}$

and $F_{\text{side}} \leq \mathbf{0}$ because we adopt the space discretization approach to achieve computational efficiency and present our formulation in the next section.

6.2 Verification in a Hierarchical Structure

The main idea of our proposed approach consists of two sequential stages. First, we formulate the safety verification problem based on (nonuniform) space discretization for better computation time. Second, we address the drawback of the space discretization approach, which is using simplified vehicle dynamics, by adopting a hierarchical structure as illustrated in Figure 6-2. In this structure, the complex dynamical vehicle model considered in the original safety verification problem (Problem 2.1) is referred to as the *concrete system*, and its coarse model as the *abstract system* or *abstraction*.

We solve the safety verification problem (Problem 2.1) by solving a verification problem of an abstraction. In particular, we construct a trajectory of the abstraction in Section 6.2.1, and find a trajectory of the concrete system that tracks the abstraction trajectory within an allowed bound in Section 6.2.2 by using a sliding mode control. This approach is applicable to by far the most general intersection geometries, such as that in Figure 2-2, including multiple side conflict areas, splitting paths, and merging paths without significantly increasing computational complexity.

6.2.1 Verification on the Abstract System

In this section, we formulate the safety verification problem based on an abstraction of the concrete system (2.4) and on space discretization. Suppose that e_{max} is an allowed maximum error between trajectories of the abstract and concrete systems. To ensure collision avoidance of the concrete system, we formulate the abstraction-based safety verification problem in terms of an inflated bad set. That is, let

$$d^* := d + 2e_{\text{max}} \text{ and } (\alpha_{i,j}^*, \beta_{i,j}^*) := (\alpha_{i,j} - e_{\text{max}}, \beta_{i,j} + e_{\text{max}}), \quad (6.4)$$

which indicates that d^* and $(\alpha_{i,j}^*, \beta_{i,j}^*)$ are the safe distance and the location of side conflict area i inflated by e_{\max} , respectively.

Abstraction

We define an abstraction of the concrete system (2.4) as a first-order dynamical system with a smoothing constant ε_j :

$$\dot{x}_{a,j} = v_j, \quad v_j \in \mathcal{V}_j(\varepsilon_j) \quad (6.5)$$

where $x_{a,j}$ is the position. Also, $v_j \in \mathcal{V}_j(\varepsilon_j)$ is a piecewise constant speed whose values are bounded between $[v_{j,\min}, v_{j,\max}] = \dot{X}_j$, and the difference between consecutive values of v_j is limited by ε_j .

Path Discretization

We divide the longitudinal path of vehicle j into N_j short segments of varying, but predetermined, lengths. Given an initial position $\mathbf{x}_0 = (x_{0,1}, \dots, x_{0,n})$, the discretized longitudinal path is denoted by a finite sequence of intervals

$$\{[\xi_j[k-1], \xi_j[k]]\}_{k=1}^{N_j}$$

where $[\xi_j[k-1], \xi_j[k]] \subset X_j$, $\xi_j[k-1] < \xi_j[k]$, $\xi_j[0] = x_{0,j}$, and $\xi_j[N_j] = x_{j,\max}$. That is,

$$\bigcup_{k=1}^{N_j} [\xi_j[k-1], \xi_j[k]] = [x_{0,j}, x_{j,\max}].$$

Let $\Delta\xi_j : \{1, 2, \dots, N_j\} \rightarrow \mathbb{R}_+$ denote a sequence of the segments' lengths, that is,

$$\Delta\xi_j[k] = \xi_j[k] - \xi_j[k-1].$$

The longitudinal path of vehicle j can be discretized such that it satisfies the following specifications, which enable simple and exact description of the collision avoidance constraints.

- 1) The end points of segments coincide with the begin or end points of rear-end conflict areas. If $(j, j') \in \mathcal{O}$, there exist positive integers k_1 and k_2 such that $[\xi_j[k_1], \xi_j[k_2]] = O_{j,j'}$;
- 2) The end points of segments coincide with the begin or end points of inflated side conflict areas. If $(j, j') \in \mathcal{D}_i$, there exist positive integers k_1 and k_2 such that $\xi_j[k_1] = \alpha_{i,j}^*$ and $\xi_j[k_2] = \beta_{i,j}^*$;
- 3) The length of some segments in rear-end conflict areas is the same as the inflated safe distance d^* . If vehicle j precedes vehicle j' , there exists a positive integer δ such that $(\xi_j[k] - d^*) - o_{j,j'} = \xi_{j'}[\delta] - o_{j',j}$ if $\xi_j[k], \xi_j[k] - d^* \in O_{j,j'}$;
- 4) Segments in rear-end conflict areas share a common discretization grid in the sense that $(\xi_j[k] - d^*) - o_{j,j'} = \xi_{j'}[\delta] - o_{j',j}$ implies $\Delta\xi_j[k+1] = \Delta\xi_{j'}[\delta+1]$ if $\xi_j[k+1], \xi_j[k+1] - d^* \in O_{j,j'}$.

Given an initial position \mathbf{x}_0 , specifications 1, 3, and 4 are necessary if $x_{0,j} \leq \sup O_{j,j'}$, which means that vehicle j has not exited the rear-end conflict area. In particular, in specification 1, if $x_{0,j} \in O_{j,j'}$, then there exist positive integers k_1 and k_2 such that $\xi_j[k_1] = x_{0,j}$ and $\xi_j[k_2] = \sup O_{j,j'}$. Similarly, in specification 2, if $x_{0,j} \in [\alpha_{i,j}^*, \beta_{i,j}^*]$, then there exist positive integers k_1 and k_2 such that $\xi_j[k_1] = x_{0,j}$ and $\xi_j[k_2] = \beta_{i,j}^*$.

The decision variables in the problem are times necessary for a vehicle to cross each space segment. That is,

$$s_j[k] = \Delta\mathbf{t}_j[k]$$

for all $j \in \{1, 2, \dots, n\}$ and $k \in \{1, 2, \dots, N_j\}$ where $\Delta\mathbf{t}_j[k]$ is the time for vehicle j to travel the distance $\Delta\xi_j[k]$.

Constraint 1: $F_{\text{dynamics}}(\mathbf{s}) \leq 0$

The abstraction (6.5) enables rewriting of the decision variable as

$$\Delta\mathbf{t}_j[k] = \frac{\Delta\xi_j[k]}{v_j},$$

if v_j is a constant speed on the k -th segment. The fact that the speed v_j is bounded by $[v_{j,\min}, v_{j,\max}]$ gives the bounds of time interval $\Delta \mathbf{t}_j$,

$$\frac{\Delta \xi_j[k]}{v_{j,\max}} \leq \Delta \mathbf{t}_j[k] \leq \frac{\Delta \xi_j[k]}{v_{j,\min}} \text{ for all } k \in \{1, 2, \dots, N_j\}. \quad (6.6)$$

If $v_{j,\min} = 0$, the constraint becomes $\Delta \xi_j[k]/v_{j,\max} \leq \Delta \mathbf{t}_j[k] < \infty$.

We also impose a smoothing constraint in the abstraction (6.5) to prevent extreme speed changes between two segments. This constraint is required to maintain a bounded error between trajectories of the abstract and concrete systems:

$$\left| \frac{\Delta \xi_j[k-1]}{\Delta \mathbf{t}_j[k-1]} - \frac{\Delta \xi_j[k]}{\Delta \mathbf{t}_j[k]} \right| = \frac{|\Delta \xi_j[k-1] \cdot \Delta \mathbf{t}_j[k] - \Delta \xi_j[k] \cdot \Delta \mathbf{t}_j[k-1]|}{\Delta \mathbf{t}_j[k-1] \Delta \mathbf{t}_j[k]} \leq \varepsilon_j.$$

We introduce a *smoothing function* $h(\Delta \mathbf{t}_j[k])$ such that $h(\Delta \mathbf{t}_j[k]) \leq \varepsilon_j \Delta \mathbf{t}_j[k-1] \Delta \mathbf{t}_j[k]$ and

$$|\Delta \xi_j[k-1] \cdot \Delta \mathbf{t}_j[k] - \Delta \xi_j[k] \cdot \Delta \mathbf{t}_j[k-1]| \leq h(\Delta \mathbf{t}_j[k]), \quad (6.7)$$

for all $k \in \{2, \dots, N_j\}$ for all $j \in \{1, 2, \dots, n\}$. For $k = 1$, set

$$|\dot{x}_{0,j} \Delta \mathbf{t}_j[1] - \Delta \xi_j[1]| \leq \varepsilon_j \Delta \mathbf{t}_j[1] := h(\Delta \mathbf{t}[1]),$$

where $\dot{\mathbf{x}}_0 = (\dot{x}_{0,1}, \dots, \dot{x}_{0,n})$ is a given initial speed. The smoothing function is related to the maximum tracking error and the control input bounds, and the determination of the function will be described in Section 6.2.2. Since we want to have a linear constraint with respect to the decision variable $\Delta \mathbf{t}_j[k]$, one favorable option is to set $h(\Delta \mathbf{t}_j[k]) = c_1 \Delta \mathbf{t}_j[k] + c_2$ for some constants c_1 and c_2 .

Constraint 2: $F_{\text{rear-end}}(\mathbf{s}) \leq 0$

Given a step K where $\xi_j[K] \in O_{j,j'}$, let a positive integer $\delta(K)$ satisfy

$$(\xi_j[K] - d^*) - o_{j,j'} = \xi_{j'}[\delta(K)] - o_{j',j}.$$

The integer $\delta(K)$ exists if $(\xi_j[K] - d^*) \in O_{j,j'}$ by the construction of the discretized paths. Since rear-end conflict area segments share a common grid, $\delta(K)$ and $\delta(K+1)$ are consecutive segments of vehicle j' .

The rear-end collision avoidance constraint is as follows. If $(j, j') \in \mathcal{O}$, and

$$\text{if vehicle } j \text{ precedes vehicle } j', \quad \sum_{k=1}^K \Delta \mathbf{t}_j[k] \leq \sum_{k=1}^{\delta(K)} \Delta \mathbf{t}_{j'}[k], \quad (6.8)$$

for all $K \in \{k \in \mathbb{Z}_+ : \xi_j[k] \in O_{j,j'} \text{ and } (\xi_j[k] - d^*) \in O_{j,j'}\}$. This means that after vehicle j reaches $\xi_j[K]$, vehicle j' can reach $(\xi_j[K] - d^*) - (o_{j,j'} - o_{j',j})$. Thus, $x_{a,j}$ and $x_{a,j'}$ are distance d^* apart at the end points of each segment. Even between the end points, the distance between $x_{a,j}$ and $x_{a,j'}$ is maintained above distance d^* because trajectories of the abstract system (6.5) are (piecewise) linear.

Constraint 3: $F_{\text{side}}(\mathbf{s}) \leq 0$

Let positive integers $K_{i,j}^{\text{in}}$ and $K_{i,j}^{\text{out}}$ denote

$$\xi_j[K_{i,j}^{\text{in}}] = \alpha_{i,j}^*, \quad \xi_j[K_{i,j}^{\text{out}}] = \beta_{i,j}^*,$$

which means that the path from the $K_{i,j}^{\text{in}}$ -th segment to the $K_{i,j}^{\text{out}}$ -th segment exactly overlaps with the inflated side conflict area $(\alpha_{i,j}^*, \beta_{i,j}^*)$. Such integers exist by the construction of the discretized paths. These allow rewriting of the expression $x_j(t) \in (\alpha_{i,j}^*, \beta_{i,j}^*)$ in terms of the decision variable as follows:

$$t \in \left(\sum_{k=1}^{K_{i,j}^{\text{in}}} \Delta \mathbf{t}_j[k], \sum_{k=1}^{K_{i,j}^{\text{out}}} \Delta \mathbf{t}_j[k] \right).$$

To avoid a side collision, two vehicles never meet inside the same side conflict area. The side collision avoidance constraint is

$$\sum_{k=1}^{K_{i,j}^{\text{out}}} \Delta \mathbf{t}_j[k] \leq \sum_{k=1}^{K_{i,j'}^{\text{in}}} \Delta \mathbf{t}_{j'}[k] \text{ or } \sum_{k=1}^{K_{i,j'}^{\text{out}}} \Delta \mathbf{t}_{j'}[k] \leq \sum_{k=1}^{K_{i,j}^{\text{in}}} \Delta \mathbf{t}_j[k], \quad (6.9)$$

for all $(j, j') \in \mathcal{D}_i$ for all side conflict area i . That is, vehicle j' enters the side conflict area after vehicle j exits or vehicle j' exits the side conflict area before vehicle j enters. This constraint can be written by introducing binary variables to indicate the precedence constraint of vehicles, the number of which is at most $mn(n-1)$, where m is the number of side conflict areas and n is the number of vehicles. Note that the number of binary variables does not grow with a finite horizon N_j .

To sum up, our formulation of the discretized safety verification problem consists in verifying nonemptiness of the region satisfying (6.6)-(6.9).

Problem 6.1 (Discretized safety verification). Given an initial state $(\mathbf{x}_0, \dot{\mathbf{x}}_0)$, determine if there is a feasible solution $\Delta \mathbf{t}_j$ for all j to

$$\begin{aligned} & \min f(\Delta \mathbf{t}_j, \forall j) \\ & \text{subject to (6.6) - (6.9),} \end{aligned}$$

for some objective function $f(\Delta \mathbf{t}_j, \forall j)$.

Trajectory of the Abstraction

Given a solution $\Delta \mathbf{t}_j$ for all j to Problem 6.1, we construct a trajectory of the abstract system, denoted by $(x_{a,j}, \dot{x}_{a,j})$, such that for all $k \in \{1, \dots, N_j\}$,

$$\begin{aligned} & \text{for } t \in \left[\sum_{i=1}^{k-1} \Delta \mathbf{t}_j[i], \sum_{i=1}^k \Delta \mathbf{t}_j[i] \right), \\ & \dot{x}_{a,j}(t) = \frac{\Delta \xi_j[k]}{\Delta \mathbf{t}_j[k]}, \quad x_{a,j}(t) = \xi_j[k-1] + \dot{x}_{a,j}(t) \cdot \left(t - \sum_{i=1}^{k-1} \Delta \mathbf{t}_j[i] \right). \end{aligned} \tag{6.10}$$

The trajectory $(x_{a,j}, \dot{x}_{a,j})$ is piecewise affine as illustrated in Figure 6-3, which is feasible for the abstraction but not for the concrete system. By considering this trajectory as a reference, we find a trajectory of the concrete system that tracks $(x_{a,j}, \dot{x}_{a,j})$ within a given maximum error e_{\max} .

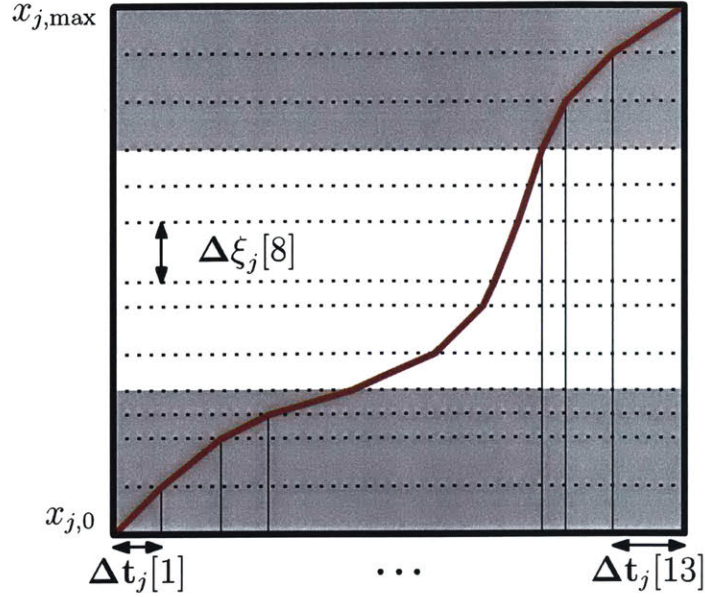


Figure 6-3: Discretized path $\{[\xi_j[k-1], \xi_j[k]]\}_{k=1}^{13}$ of vehicle j and sequence of time intervals Δt_j , based on which a trajectory of the abstraction (red line) is defined.

6.2.2 Verification on the Concrete System

This section describes the verification of the concrete system (Problem 2.1) given a reference $(\mathbf{x}_a, \dot{\mathbf{x}}_a)$ constructed by the map (6.10). As the method of designing the interface between the abstract and concrete systems, we employ sliding mode control [54], which guarantees safety of vehicles under some conditions.

Sliding Mode Control

In this section, we focus on an input signal of an individual vehicle, and thus, omit the subscript j for notational simplicity, which has been used to indicate vehicle j .

As an indicator of tracking errors, a scalar function, $s : X \times \dot{X} \rightarrow \mathbb{R}$, is defined such that

$$s(x, \dot{x}) := (\dot{x} - \dot{x}_a) + \lambda(x - x_a),$$

where $\lambda > 0$ is a design parameter, and (x_a, \dot{x}_a) is a given reference trajectory that the state (x, \dot{x}) of the concrete system needs to track. The goal is to design a control input that maintains s close to zero. To do this, a control input is designed such that

if $|s(x, \dot{x})| > \Phi$ for some $\Phi > 0$,

$$\frac{1}{2} \frac{d}{dt} |s(x, \dot{x})|^2 \leq -\eta |s(x, \dot{x})|, \quad (6.11)$$

where $\eta > 0$ is a design parameter. The control input satisfying (6.11) drives the state into a thin boundary layer $S = \{(x, \dot{x}) : |s(x, \dot{x})| \leq \Phi\}$. Once the state enters, it stays inside the boundary layer unless there is an extreme change in the reference trajectory that makes $|s(x, \dot{x})|$ greater than Φ .

We use the following input, which satisfies (6.11) with equality if letting $\dot{x}_a \equiv 0$ (by neglecting a finite number of discontinuities of \dot{x}_a),

$$u = \frac{1}{b(x, \dot{x})} \left(-f(x, \dot{x}) - \lambda(\dot{x} - \dot{x}_a) - \eta \operatorname{sat} \left(\frac{s(x, \dot{x})}{\Phi} \right) \right) \quad (6.12)$$

where $\operatorname{sat}(q)$ is 1 if $q > 1$, -1 if $q < -1$, and q otherwise. Typically, the sliding mode control input makes the state converge to the sliding surface $\{(x, \dot{x}) : |s(x, \dot{x})| = 0\}$ by considering $\Phi = 0$, but such control input causes a chattering behavior of the state. To avoid an issue of chattering, we consider the boundary layer with thickness of strictly positive Φ .

One property of the sliding mode control is that it provides quantified upper bounds of tracking errors, $|x - x_a|$ and $|\dot{x} - \dot{x}_a|$. If $|s(x(t), \dot{x}(t))| \leq S$ for all $t \geq 0$, we have

$$|x(t) - x_a(t)| \leq \frac{S}{\lambda}, \quad |\dot{x}(t) - \dot{x}_a(t)| \leq 2S \quad (6.13)$$

for all $t \geq 0$.

Interface between the Abstract and Concrete Systems

The parameters λ, η , and Φ play an important role in determining the tracking error bounds. In the following, we provide some conditions of λ, η, Φ that are necessary to ensure the tracking error $|x - x_a|$ bounded by e_{\max} . Let the parameters Φ and λ be constant over all segments, and the parameter η be constant at η_k on the k -th

segment and vary between segments. Let $\Delta \mathbf{v}[k]$ denote the speed difference between two consecutive segments $k - 1$ and k of the abstraction trajectory, that is,

$$\Delta \mathbf{v}[k] := \frac{\Delta \xi[k-1]}{\Delta \mathbf{t}[k-1]} - \frac{\Delta \xi[k]}{\Delta \mathbf{t}[k]}.$$

For $k = 1$, set $\Delta \mathbf{v}[1] := \dot{x}_0 - \Delta \xi[1]/\Delta \mathbf{t}[1]$.

Condition 6.1. The design parameters η and λ satisfy the following:

- $\eta = \eta_k \geq |\Delta \mathbf{v}[k]|/\Delta \mathbf{t}[k]$ on k -th segment;
- $\lambda \geq (\Phi + \max_k |\Delta \mathbf{v}[k]|)/e_{\max}$.

We prove the lemmas that Condition 6.1 ensures the position tracking error bounded by e_{\max} .

Lemma 6.1. *If η and λ satisfy Condition 6.1, the tracking error $|x(t) - x_a(t)|$ is bounded by e_{\max} for all $t \in [0, \sum_{k=1}^N \Delta \mathbf{t}[k]]$.*

Proof. We will prove that $|s(x, \dot{x})| \leq \Phi + |\Delta \mathbf{v}[k]|$ on each k -th segment by mathematical induction on k . On the first segment, $|s(x, \dot{x})| \leq |\Delta \mathbf{v}[1]|$ because $x_0 = x_a(0)$, $|\dot{x}(0) - \dot{x}_a(0)| = |\Delta \mathbf{v}[1]|$, and $|s(x, \dot{x})|$ does not increase in time on the segment by (6.11). Suppose $|s(x, \dot{x})| \leq \Phi + |\Delta \mathbf{v}[k-1]|$ on the $(k-1)$ -th segment. Then, since $\eta_{k-1} \geq |\Delta \mathbf{v}[k-1]|/\Delta \mathbf{t}[k-1]$ and $|s(x, \dot{x})|$ linearly decreases in time at rate η_{k-1} , we have $|s(x, \dot{x})| \leq \Phi$ at the end of the $(k-1)$ -th segment. Because between the $(k-1)$ -th and k -th segments, the speed of the abstraction trajectory changes by $\Delta \mathbf{v}[k]$ and the position does not change instantaneously, $|s(x, \dot{x})|$ increases by at most $|\Delta \mathbf{v}[k]|$, that is, $|s(x, \dot{x})| \leq \Phi + |\Delta \mathbf{v}[k]|$. Since $|s(x, \dot{x})|$ does not increase in time on a segment, $|s(x, \dot{x})| \leq \Phi + |\Delta \mathbf{v}[k]|$ on the k -th segment. This concludes the proof because by (6.13)

$$|x(t) - x_a(t)| \leq \frac{\Phi + |\Delta \mathbf{v}[k]|}{\lambda} \quad \text{for } t \in \left[\sum_{i=1}^{k-1} \Delta \mathbf{t}[i], \sum_{i=1}^k \Delta \mathbf{t}[i] \right),$$

which becomes $|x(t) - x_a(t)| \leq e_{\max}$ since $\lambda \geq (\Phi + \max_k |\Delta \mathbf{v}[k]|)/e_{\max}$ by Condition 6.1. \square

However, the parameters η and λ cannot be made arbitrarily large to satisfy Condition 6.1 because the magnitude of control input (6.12) increases with respect to η and λ , and should be bounded by $[u_{\min}, u_{\max}]$. By appropriately restricting speed changes of the abstraction trajectory, we can enforce the input bound constraint. In the following, we provide a condition for the smoothing function h , which is part of constraint (6.7) in Problem 6.1.

Condition 6.2. The smoothing function h at the k -th segment is denoted by h_k , which is a map from $[\Delta\xi[k]/v_{\max}, \Delta\xi[k]/v_{\min}]$ to \mathbb{R}_+ , such that, with $\Delta\mathbf{t}[k-1]$ satisfying $|\Delta\xi[k-1] \cdot \Delta\mathbf{t}[k] - \Delta\xi[k] \cdot \Delta\mathbf{t}[k-1]| \leq h_k(\Delta\mathbf{t}[k])$,

$$u_{\min} \leq \frac{1}{b(x, \dot{x})} \left(-f(x, \dot{x}) - \frac{2}{e_{\max}} \left(\Phi + \max_{k, \Delta\mathbf{t}[k]} \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]} \right)^2 - \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]^2} \right),$$

$$u_{\max} \geq \frac{1}{b(x, \dot{x})} \left(-f(x, \dot{x}) + \frac{2}{e_{\max}} \left(\Phi + \max_{k, \Delta\mathbf{t}[k]} \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]} \right)^2 + \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]^2} \right),$$

for all $\Delta\mathbf{t}[k] \in [\Delta\xi[k]/v_{\max}, \Delta\xi[k]/v_{\min}]$.

Since the smoothing function h should be determined before solving Problem 6.1, it must satisfy the inequalities in Condition 6.2 for all possible solutions. The set of smoothing functions h that satisfy Condition 6.2 is nonempty because if $h_k \equiv 0$ for all k ,

$$u_{\min} \leq \frac{1}{b(x, \dot{x})} \left(-f(x, \dot{x}) - \frac{2\Phi^2}{e_{\max}} \right), \quad u_{\max} \geq \frac{1}{b(x, \dot{x})} \left(-f(x, \dot{x}) + \frac{2\Phi^2}{e_{\max}} \right),$$

which are true because e_{\max} is nonzero, Φ is arbitrarily small, and $-f(x, \dot{x})/b(x, \dot{x}) \in (u_{\min}, u_{\max})$ for all $x \in X, \dot{x} \in \dot{X}$ by assumption. Therefore, $h \equiv 0$ is in the set.

Lemma 6.2. *Suppose that $\Delta\mathbf{t}$ is a solution to Problem 6.1 based on the smoothing function h satisfying Condition 6.2. Then there exist the design parameters η and λ that satisfy Condition 6.1 and make the control input (6.12) bounded by $[u_{\min}, u_{\max}]$. Also, the control input yields a trajectory of the concrete system that deviates no more than e_{\max} from the abstraction trajectory defined by $\Delta\mathbf{t}$.*

Proof. Condition 6.1 is satisfied if

$$\eta_k = \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]^2} \text{ and } \lambda = \frac{1}{e_{\max}} \left(\Phi + \max_{k, \Delta\mathbf{t}[k]} \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]} \right)$$

because from constraint (6.7),

$$|\Delta\mathbf{v}[k]| \leq \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}([k-1])\Delta\mathbf{t}[k]}.$$

Thus, by Lemma 6.1, the tracking error $|x - x_a|$ is bounded by e_{\max} , and $|\dot{x} - \dot{x}_a| \leq 2e_{\max}\lambda$. Then the control input (6.12) becomes the right-hand side terms in Condition 6.2, which implies that the input is bounded by $[u_{\min}, u_{\max}]$. \square

An analytic expression of h satisfying Condition 6.2 can be obtained before solving Problem 6.1. Let $\mathbf{g}_k(h_k, \Delta\mathbf{t}[k], \Delta\mathbf{v}_{\max}) \leq 0$ denote the inequalities in Condition 6.2 in which $b(x, \dot{x})$, $f(x, \dot{x})$, $\Delta\mathbf{t}[k-1]$ are substituted with their worst-case bounds and $\max_{k, \Delta\mathbf{t}[k]} h_k(\Delta\mathbf{t}[k]) / (\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k])$ by $\Delta\mathbf{v}_{\max}$. First, assume $\Delta\mathbf{v}_{\max}$ is 0 and find h_k as a function of $\Delta\mathbf{t}[k]$ so that it satisfies the inequalities $\mathbf{g}_k(h_k, \Delta\mathbf{t}[k], 0) \leq 0$. With the obtained h_k , compute $\max_{k, \Delta\mathbf{t}[k]} h_k(\Delta\mathbf{t}[k]) / (\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k])$. If this maximum value is greater than the value of $\Delta\mathbf{v}_{\max}$, let $\Delta\mathbf{v}_{\max}$ take the maximum value and find again h_k that satisfies $\mathbf{g}_k(h_k, \Delta\mathbf{t}[k], \Delta\mathbf{v}_{\max}) \leq 0$. If the maximum value is not greater than the value of $\Delta\mathbf{v}_{\max}$ used to find h_k , then stop the iteration. The following example shows this process.

Example 6.1. Consider the vehicle dynamics $\ddot{x} = -0.005\dot{x}^2 + u$, which is the same as (2.5) with $c_1 = 0.005$, $c_2 = 0$, $c_3 = 1$. Suppose $\Delta\xi[k] = 3$ for all k , $e_{\max} = 1$, $[v_{\min}, v_{\max}] = [1, 15]$, $[u_{\min}, u_{\max}] = [-3, 3]$, and $\Phi = 0.001$. We want to find linear smoothing functions $h_k(\Delta\mathbf{t}[k]) = c_1\Delta\mathbf{t}[k] + c_2$ that satisfy Condition 6.2.

First, suppose that the speed decreases between the $(k-1)$ -th and the k -th segments, in which case the lower bound of the control input is important to ensure. By denoting the maximum speed change by $\Delta\mathbf{v}_{\max}$,

$$-3 \leq 0.005\dot{x}^2 - 2(0.001 + \Delta\mathbf{v}_{\max})^2 - \frac{h_k(\Delta\mathbf{t}[k])}{\Delta\mathbf{t}[k-1]\Delta\mathbf{t}[k]^2}. \quad (6.14)$$

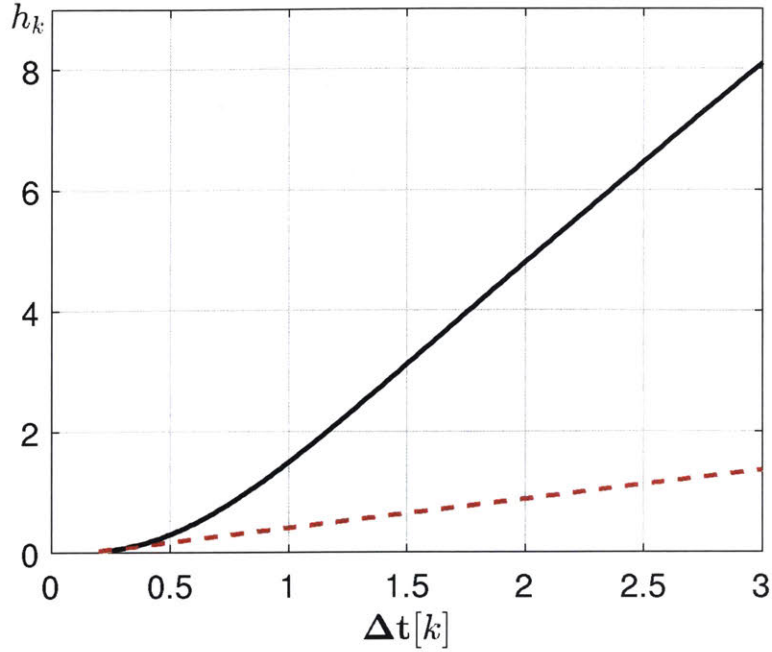


Figure 6-4: With $\Delta \mathbf{v}_{\max} = 0$, the smoothing function that satisfies (6.15) with equality is represented by the black solid line, and a linear smoothing function that satisfies (6.15) (and thus Condition 6.2) by the red dotted line.

From the smoothing constraint (6.7), $\Delta \xi[k-1] \cdot \Delta \mathbf{t}[k] - \Delta \xi[k] \cdot \Delta \mathbf{t}[k-1] \leq h_k(\Delta \mathbf{t}[k])$, thereby implying $\Delta \mathbf{t}[k-1] \geq \Delta \mathbf{t}[k] - \frac{1}{3}h_k(\Delta \mathbf{t}[k])$ in the decelerating case. Also, we have $\dot{x} \geq \Delta \xi[k]/\Delta \mathbf{t}[k] - 2(\Phi + \Delta \mathbf{v}_{\max})$ on the k -th segment because \dot{x} decreases over the segment and satisfies $|\dot{x} - \dot{x}_a| \leq 2(\Phi + \Delta \mathbf{v}_{\max})$ where $\dot{x}_a = \Delta \xi[k]/\Delta \mathbf{t}[k]$ at the end of the segment. In (6.14), we substitute \dot{x} with $\Delta \xi[k]/\Delta \mathbf{t}[k] - 2(\Phi + \Delta \mathbf{v}_{\max})$, and $\Delta \mathbf{t}[k-1]$ with $\Delta \mathbf{t}[k] - \frac{1}{3}h_k(\Delta \mathbf{t}[k])$.

$$-3 \leq 0.005 \left(\frac{3}{\Delta \mathbf{t}[k]} - 2(0.001 + \Delta \mathbf{v}_{\max}) \right)^2 - 2(0.001 + \Delta \mathbf{v}_{\max})^2 - \frac{h_k(\Delta \mathbf{t}[k])}{(\Delta \mathbf{t}[k] - \frac{1}{3}h_k)\Delta \mathbf{t}[k]} \quad (6.15)$$

Assume $\Delta \mathbf{v}_{\max} = 0$. The smoothing function h_k that satisfies (6.15) with equality is depicted as a solid black line in Figure 6-4. Any function that lies below this black line satisfies (6.15). In the figure, the red dotted line represents the linear smoothing function $0.4761\Delta \mathbf{t}[k] - 0.0751$, which satisfies (6.15), is non-negative for all $\Delta \mathbf{t}[k] \in [\Delta \xi[k]/v_{\max}, \Delta \xi[k]/v_{\min}]$, and maximizes the trapezoidal area below the

linear function. Similarly for an increasing speed, $h_k(\Delta\mathbf{t}[k]) = 0.4560\Delta\mathbf{t}[k] - 0.0777$ is the linear smoothing function that satisfies Condition 6.2, is non-negative over the domain, and maximizes the trapezoidal area below the function. With these two linear smoothing functions, the maximum speed change is 0.8217. With $\Delta\mathbf{v}_{\max} = 0.8217$, we again obtain $h_k = 0.2708\Delta\mathbf{t}[k] - 0.0429$ for the decelerating case and $0.1958\Delta\mathbf{t}[k] - 0.0354$ for the accelerating case. These new smoothing functions yield the maximum speed change of 0.4474, which is smaller than the value (0.8217) used to find the functions. Thus, the constraint (6.7) becomes

$$\begin{aligned} \Delta\xi[k-1] \cdot \Delta\mathbf{t}[k] - \Delta\xi[k] \cdot \Delta\mathbf{t}[k-1] &\leq 0.2708\Delta\mathbf{t}[k] - 0.0429, \\ -\Delta\xi[k-1] \cdot \Delta\mathbf{t}[k] + \Delta\xi[k] \cdot \Delta\mathbf{t}[k-1] &\leq 0.1958\Delta\mathbf{t}[k] - 0.0354. \end{aligned}$$

Given that Conditions 6.1 and 6.2 are satisfied, we prove that our approach guarantees that the existence of the solution to Problem 6.1 implies the existence of the solution to Problem 2.1.

Theorem 6.1. *Given a state $(\mathbf{x}_0, \dot{\mathbf{x}}_0)$, if Problem 6.1 has a feasible solution, then Problem 2.1 has a feasible solution $\mathbf{u} \in \mathcal{U}$.*

Proof. A sequence of time intervals that is feasible in Problem 6.1 is used to construct an abstraction trajectory, based on which the sliding mode control input is computed according to (6.12). This input signal makes the concrete system's state track the abstraction trajectory within an error smaller than or equal to e_{\max} , given that the design parameters satisfy Condition 6.1. Since d^* and $(\alpha_{i,j}^*, \beta_{i,j}^*)$ used in Problem 6.1 are d and $(\alpha_{i,j}, \beta_{i,j})$ inflated by e_{\max} , respectively, the resulting trajectory ensures that a safe distance d is maintained and that two vehicles never meet inside a side conflict area. Therefore, the sliding mode control input satisfies (2.7). \square

Corollary 6.1. *If $(\mathbf{x}_a, \dot{\mathbf{x}}_a)$ is a trajectory of the abstraction (6.10) generated by the map (6.10) using a solution to Problem 6.1, the solution $\mathbf{u} \in \mathcal{U}$ to Problem 2.1 is computed by an algebraic mapping (6.12).*

The converse of Theorem 6.1 is not always true, that is, a solution to Problem 2.1

can exist while no solution exists to Problem 6.1. This implies that the supervisor based on the solution to Problem 6.1 is more restrictive than the least restrictive supervisor.

6.3 Supervisor Algorithm

We design the supervisor based on the solution to Problem 6.1. We denote by VERIFICATION the procedure that takes $(\mathbf{x}_0, \dot{\mathbf{x}}_0)$ as an input and returns the solution $(\Delta \mathbf{t}_j, \forall j)$ to Problem 6.1 if exists. If the problem does not have a feasible solution, set $\Delta \mathbf{t}_j = \emptyset$ for all j . Also, we denote by SLIDINGMODE the procedure that takes the solution $(\Delta \mathbf{t}_j, \forall j)$ of Problem 6.1 and returns an input signal $\mathbf{u} \in \mathcal{U}$ generated by the formula (6.12).

Algorithm 6-8 Implementation of the supervisor at time step k

```

1: procedure SUPERVISOR( $\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k$ )
2:    $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}}) \leftarrow (\mathbf{x}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \dot{\mathbf{x}}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)))$ 
3:    $(\Delta \mathbf{t}_j, \forall j) \leftarrow \text{VERIFICATION}(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$ 
4:   if  $\Delta \mathbf{t}_j \neq \emptyset, \forall j$  then
5:      $\mathbf{u}_s^k \leftarrow \text{SLIDINGMODE}(\Delta \mathbf{t}_j, \forall j)$ 
6:     return  $\mathbf{u}_d^k$ 
7:   else
8:      $(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}}) \leftarrow (\mathbf{x}(\tau, \mathbf{u}_s^{k-1}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \dot{\mathbf{x}}(\tau, \mathbf{u}_s^{k-1}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)))$ 
9:      $(\Delta \mathbf{t}'_j, \forall j) \leftarrow \text{VERIFICATION}(\mathbf{x}_{\text{next}}, \dot{\mathbf{x}}_{\text{next}})$ 
10:    if  $\Delta \mathbf{t}'_j \neq \emptyset$  then
11:       $\mathbf{u}_s^k \leftarrow \text{SLIDINGMODE}(\Delta \mathbf{t}'_j, \forall j)$ 
12:    else
13:       $\mathbf{u}_s^k \leftarrow \mathbf{u}_s^{k-1}$  restricted to time  $[\tau, \infty)$ 
14:    return  $\mathbf{u}_s^{k-1}$  restricted to time  $[0, \tau)$ 

```

The supervisor takes as inputs the current state $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ and the desired input \mathbf{u}_d , based on which the one-step ahead state is defined (line 2). If there is a

solution to Problem 6.1 (line 4), which implies by Theorem 6.1 that an input signal that avoids the bad set exists, then the supervisor stores the safe input signal \mathbf{u}_s^k for a possible use at future steps (line 5) and allows the desired input (line 6). If Problem 6.1 is infeasible, the supervisor neglects the desired input and returns the safe input signal stored at the previous step (line 14). To update the safe input signal, Problem 6.1 is solved again based on the one-step ahead state determined by the safe input signal at the previous step (line 8). If a feasible solution exists (line 10), a new input signal is stored as a safe input signal (line 11), and otherwise, the safe input signal stored at the previous time step \mathbf{u}_s^{k-1} is translated by τ and reused as a safe input signal (line 13).

Theorem 6.2. *Suppose initially $\text{SUPERVISOR}(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{u}_d^0)$ returns a nonempty output for some desired input $\mathbf{u}_d^0 \in \mathbf{U}$. At time $k\tau$, if $\text{SUPERVISOR}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k)$ returns \mathbf{u}_{out} , then \mathbf{u}_{out} is a safe input signal, that is,*

$$\exists \mathbf{u} \in \mathcal{U} : \mathbf{x}(t, \mathbf{u}, \mathbf{x}(\tau, \mathbf{u}_{out}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \dot{\mathbf{x}}(\tau, \mathbf{u}_{out}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))) \notin B, \forall t \geq 0. \quad (6.16)$$

Also, the procedure SUPERVISOR in Algorithm 6-8 is more restrictive than the supervisor s in (2.8) in the sense that $\mathbf{u}_{out} = \mathbf{u}_d^k$ implies $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k$.

Proof. The output of Algorithm 6-8 is either \mathbf{u}_d^k (line 6) or \mathbf{u}_s^{k-1} restricted to time $[0, \tau)$ (line 14). If $\mathbf{u}_{out} = \mathbf{u}_d^k$, there exists a solution $(\Delta \mathbf{t}_j, \forall j)$ to Problem 6.1 given a state $(\mathbf{x}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)), \dot{\mathbf{x}}(\tau, \mathbf{u}_d^k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)))$, which implies (6.16) by Theorem 6.1. If $\mathbf{u}_{out} = \mathbf{u}_s^{k-1}$ restricted to time $[0, \tau)$, (6.16) is true because the input signal \mathbf{u}_s^{k-1} restricted to time $[\tau, \infty)$ satisfies the condition in (6.16). This is because \mathbf{u}_s^{k-1} gets the value from $\text{SLIDINGMODE}(\Delta \mathbf{t}_j, \forall j)$ or $\text{SLIDINGMODE}(\Delta \mathbf{t}'_j, \forall j)$ when $\Delta \mathbf{t}_j$ or $\Delta \mathbf{t}'_j$ is nonempty, and the sliding mode control input (6.12) is the solution to Problem 2.1 by Theorem 6.1.

If $\mathbf{u}_{out} = \mathbf{u}_d^k$ given $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k)$, there is a feasible solution $\Delta \mathbf{t}_j, \forall j$ to Problem 6.1, which implies the existence of a solution to Problem 2.1 by Theorem 6.1. Thus, $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k$. \square

6.4 Validation of the Supervisor: Simulations

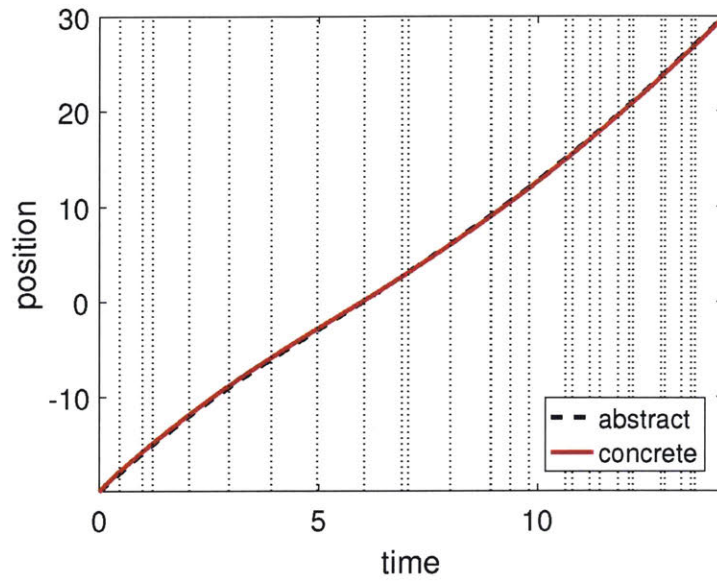
We implement Algorithm 6-8 in the traffic scenario depicted in Figure 2-2 using MATLAB on a personal computer consisting of an Intel Core i7 processor at 3.10 GHz and 8 GB RAM. We use CPLEX [33] to solve mixed integer programming problems. Also, in the scenario depicted in Figure 6-1, we perform computational experiments to compare the performance of our space discretization-based approach with the performance of the time discretization-based approach described in Section 6.1.1.

6.4.1 Implementation of the Supervisor Algorithm

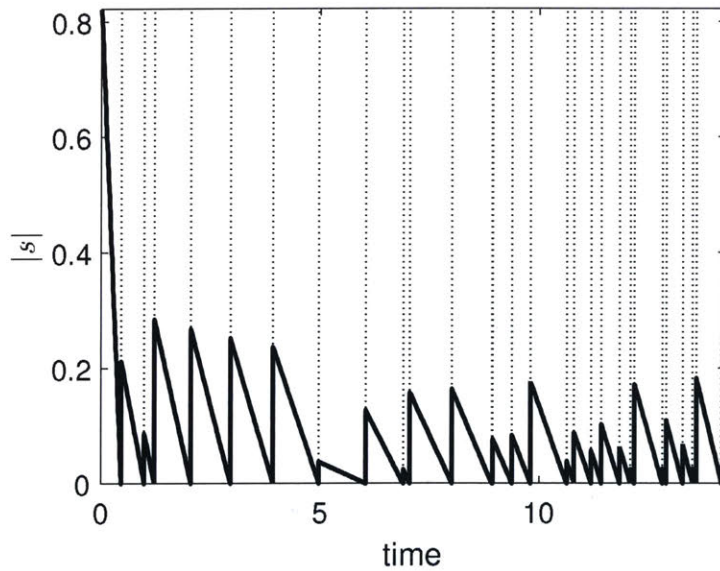
In the scenario depicted in Figure 2-2, consider the longitudinal vehicle dynamics (2.5) where $c_1 = 0.005$, $c_2 = 0$, and $c_3 = 1$. The paths of vehicles are first uniformly discretized into small segments of length $\Delta\xi_j = 3$, and then further refined to include specific locations to satisfy the specifications discussed in Section 6.2.1. We use the same values for the parameters v_{\min} , v_{\max} , u_{\min} , u_{\max} , Φ , and e_{\max} as in Example 6.1.

Given a state, we solve Problem 6.1 based on the abstract system and compute a sliding mode control (6.12) that generates a trajectory of the concrete system. Figure 6-5(a) shows that a trajectory of the concrete system tracks an abstraction trajectory within the error bound $e_{\max} = 1$, which confirms Lemma 6.1. The vertical dotted lines indicate the solution of Problem 6.1 over the discretized path of a vehicle. Figure 6-5(b) shows the magnitude of the scalar function s over time. At the beginning of each segment, $|s|$ increases by $|\Delta\mathbf{v}[k]|$ and decreases linearly over time. It becomes smaller than or equal to Φ at the end of each segment as we stated in the proof of Lemma 6.1.

Now, we implement the supervisor algorithm (Algorithm 6-8) in the scenario of Figure 2-2. The supervisor prevents side collisions among 20 vehicles on 36 side conflict areas, and rear-end collisions on splitting and merging paths. Figure 6-6 shows trajectories of vehicles crossing the intersection. The solid line represents the trajectory of vehicle 16, which travels through two rear-end conflict areas (gray regions $O_{16,7}$ and $O_{16,9}$, at the same location as $O_{16,10}$) and five side conflict areas (red regions



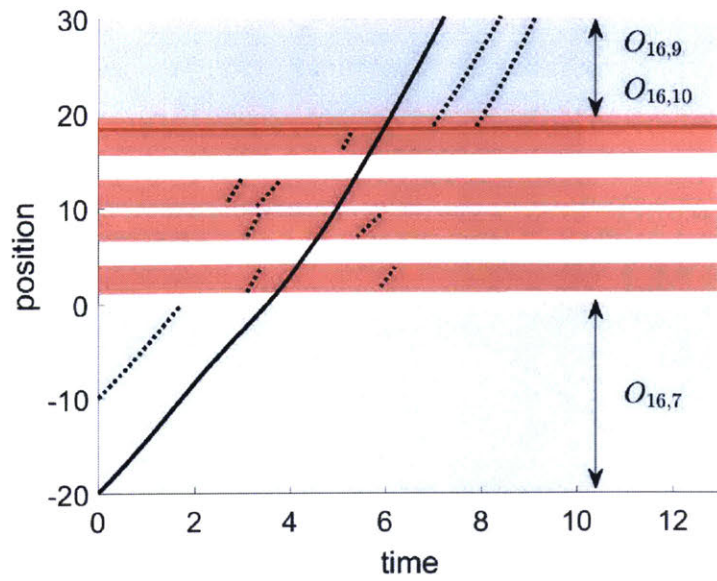
(a)



(b)

Figure 6-5: With λ, η , and h satisfying Conditions 6.1 and 6.2, a trajectory of the concrete system tracks an abstraction trajectory within $e_{\max} = 1$, and the magnitude of s decreases over a segment. The vertical dotted lines indicate time intervals taken to travel each segment.

between the positions 0 and 20). The dotted lines represent the trajectories of other vehicles that share the rear-end or side conflict areas. Note that vehicles do not meet inside a side conflict area, and vehicles maintain a safe distance of 4 on the rear-end



(a)

Figure 6-6: Trajectory of vehicle 16 through two rear-end conflict areas ($O_{16,7}$ and $O_{16,9} = O_{16,10}$) and five side conflict areas, as depicted in Figure 2-2. The dotted lines represent the trajectories of other vehicles that share the same rear-end or side conflict area. Vehicles maintain a safe distance of 4 on the rear-end conflict areas, and are not simultaneously present inside each side conflict area.

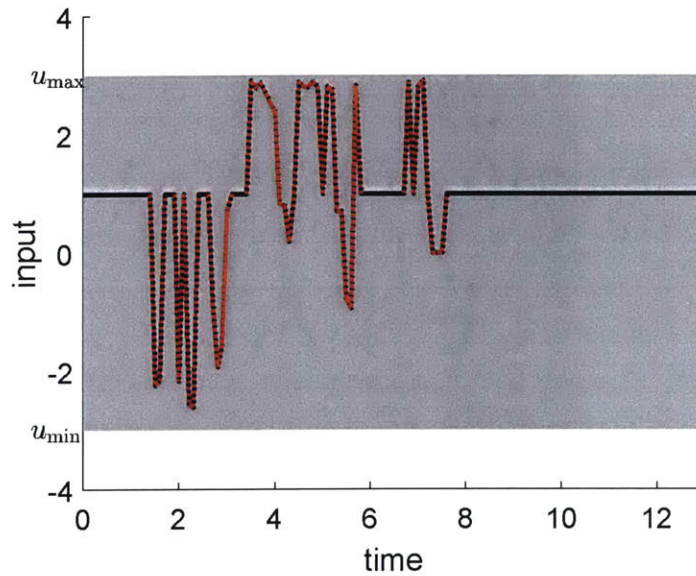


Figure 6-7: Input signal applied to vehicle 16. The supervisor overwrites the desired input (black line) when a future collision is detected. The dotted red line indicates the sliding mode control input applied by the supervisor.

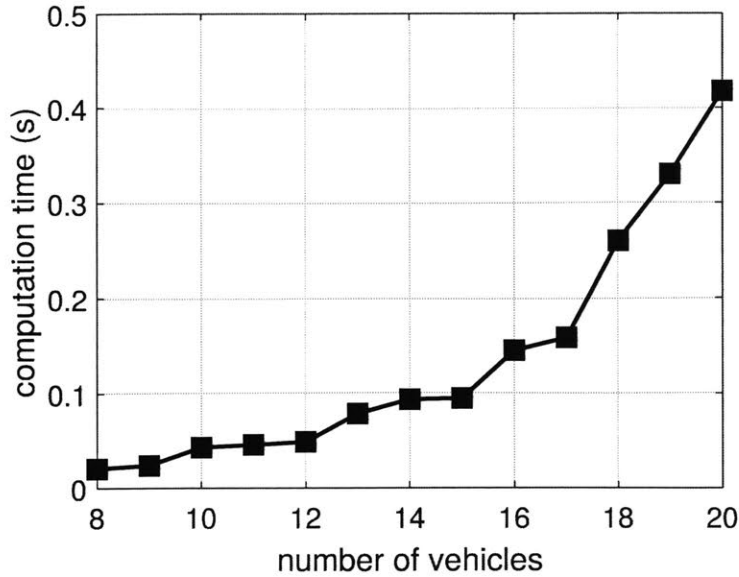


Figure 6-8: Computation time to solve Problem 6.1 as the number of vehicles in the scenario of Figure 2-2 increases.

conflict areas. The input signal applied to vehicle 16 is illustrated in Figure 6-7. The desired input is 1 (solid black line) and is overridden by the supervisor (dotted red line) when a potential collision is detected (i.e., when Problem 6.1 has no feasible solution). Note that the input signal is bounded by $[u_{\min}, u_{\max}]$ at all times because we use the control design parameters η, λ and the smoothing function h that satisfy Conditions 6.1 and 6.2, respectively. This confirms Lemma 6.2. As increasing the number of vehicles in the scenario from 8 to 20, we measured the maximum computation time to solve Problem 6.1 in Algorithm 6-8. As shown in Figure 6-8, our approach can handle realistic and complicated scenarios, such as that in Figure 2-2 with 20 vehicles, within 0.45 s.

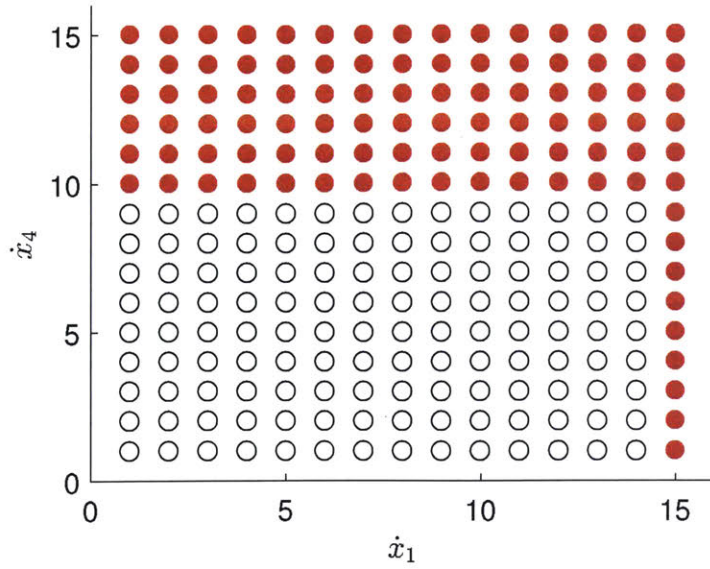
6.4.2 Performance Comparisons

In the scenario depicted in Figure 6-1, we compare the performance of our space discretization-based safety verification (Problem 6.1) with that of time discretization-based safety verification discussed in Section 6.1.1. The comparison is performed in terms of two measures: restrictiveness and computation time. If a problem finds a

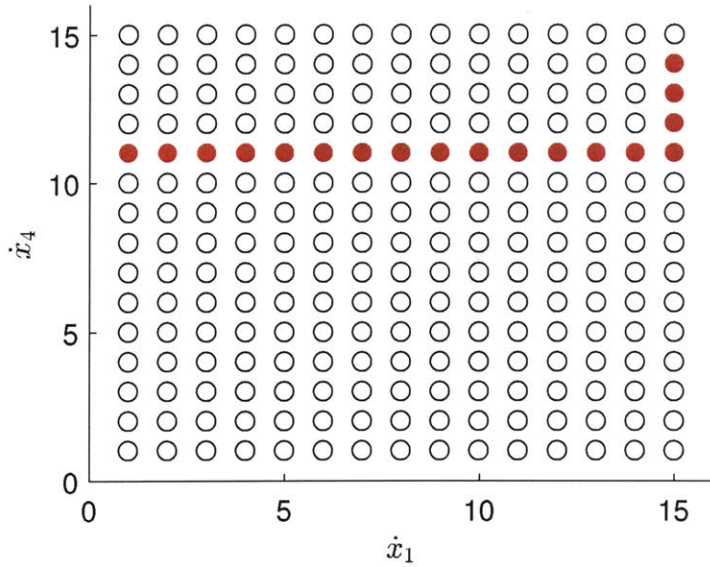
feasible solution given a state while the other problem does not, the problem is less restrictive than the other problem. For computation time, we measure the CPU time taken to solve each verification problem.

We consider linear vehicle dynamics $\ddot{x}_j = u_j$, which is the same as (2.5) with $c_1 = c_2 = 0$ and $c_3 = 1$. While our approach based on space discretization can handle nonlinearity of vehicle dynamics by using the abstraction and sliding mode control, the time discretization approach requires the assumption of linear vehicle dynamics to make the constraint $F_{\text{dynamics}} \leq 0$ linear. We let $\mathbf{x}(0) = (-15, -6, -15, -15)$ and $\dot{\mathbf{x}}(0) = (\dot{x}_1, 15, 15, \dot{x}_4)$ be an initial state, where a vehicle at position 0 indicates that the vehicle's center just enters the intersection, and determine if each problem has a feasible solution while varying \dot{x}_1 and \dot{x}_4 from $\dot{x}_{\min} = 1$ to $\dot{x}_{\max} = 15$.

Restrictiveness and computational time depend on the size of (time or space) discretization grid. If a finer discrete step is considered, the verification tends to be less restrictive but requires more computation time. In Figure 6-9, we compare the results of our approach (Problem 6.1) on two different spatial grid, where $\Delta\xi_j \approx 3$ means that we first divide the path uniformly into segments of length 3, and then refine the grid to satisfy the specifications given in Section 6.2.1. If Problem 6.1 does not have a feasible solution, we place a red dot, and otherwise, a white dot. The smaller number of red dots indicates that the approach is less restrictive with a finer spatial grid; for example, given a state with $\dot{x}_1 = 15$ and $\dot{x}_4 = 1$, Problem 6.1 with $\Delta\xi_j = 0.5$ finds a feasible solution, whereas it is not possible with $\Delta\xi_j = 3$. In both cases, the computation time is acceptable, which usually means less than 0.1 s [60]. In Figure 6-10, the time discretization-based verification problem is evaluated with two different time steps, $\Delta t = 1$ and $\Delta t = 0.15$. The problem with a smaller time step is less restrictive, but the computation time grows exponentially, because a smaller time step increases the number of binary variables as discussed in Section 6.1.1. In these simulations, 8 binary variables are involved in our space discretization-based approach, whereas 863 and 5708 binary variables are involved in the time discretization-based approach with $\Delta t = 1$ and 0.15, respectively. The results show that our approach does not lose restrictiveness compared to the time discretization-based approach while exhibiting



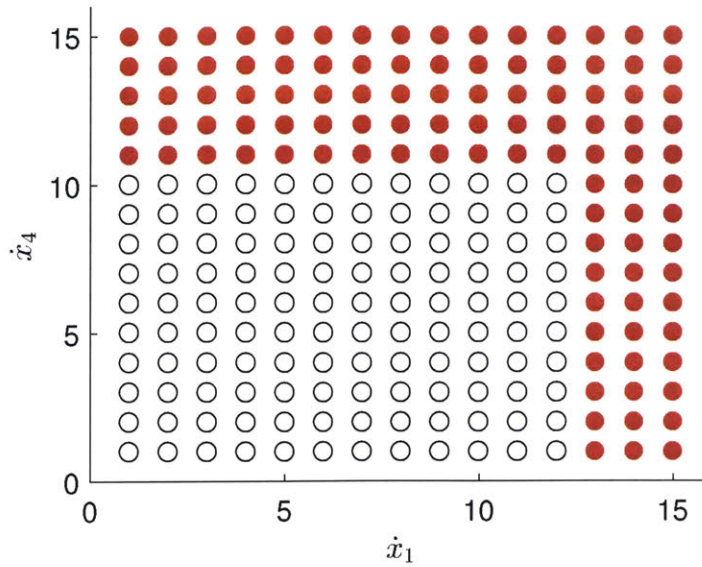
(a) $\Delta\xi_j \approx 3$, Average computation time: 0.005 s



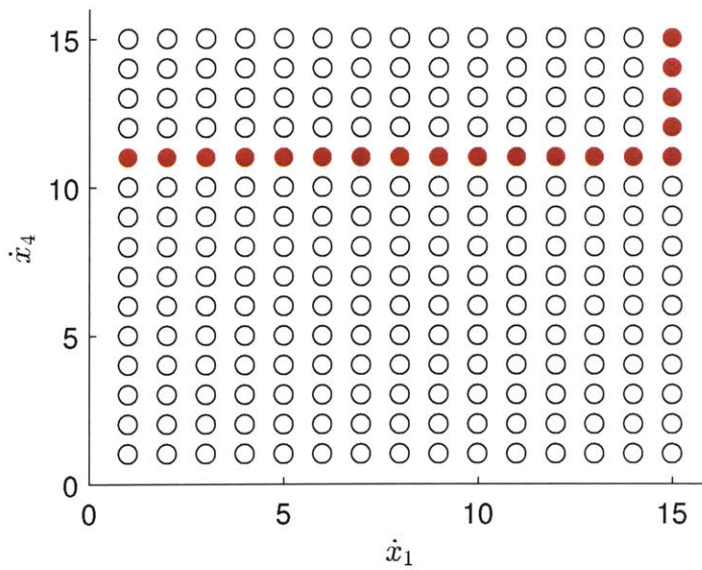
(b) $\Delta\xi_j \approx 0.5$, Average computation time: 0.076 s

Figure 6-9: As varying \dot{x}_1 and \dot{x}_4 from $\dot{x}_{\min} = 1$ to $\dot{x}_{\max} = 15$, we plot a white dot if there is a feasible solution to Problem 6.1 given a state $\mathbf{x}(0) = (-15, -6, -15, -15)$ and $\dot{\mathbf{x}}(0) = (\dot{x}_1, 15, 15, \dot{x}_4)$, and a red dot otherwise. We use $e_{\max} = 1$ in (a) and 0.25 in (b). The smaller number of red dots in (b) indicates that our problem on the finer grid is less restrictive while requiring more computation time.

favorable computation time.



(a) $\Delta t = 1$, Average computation time: 0.034s



(b) $\Delta t = 0.15$, Average computation time: 1.069s

Figure 6-10: Results of time discretization-based verification, with different time steps. Our approach in Figure 6-9 is not more restrictive than the time discretization-based verification while computationally favorable.

Chapter 7

Conclusion

This thesis presented the design of supervisor algorithms in three intersection models of different complexity levels. The supervisors determine the existence of a future collision among vehicles at or near an intersection (safety verification) and override the drivers with a safe input if a future collision is detected. We translated the safety verification problem into scheduling problems by exploiting the monotonicity of the system. Since scheduling problems are combinatorial with the number of vehicles, we approximately solved the problems within quantified approximation error bounds. We designed the supervisors based on the (exact or approximate) solutions to the scheduling problems, proved that they ensure safety and are nonblocking, and validated them through computer simulations and experiments. In particular, computer simulations showed that the supervisors could be implemented in real-time for most scenarios of realistic size.

In more detail, in Chapter 4, when modeling an intersection as a single conflict area, we considered several sources of uncertainty, such as errors in the dynamical model and measurement and the presence of uncontrolled vehicles, and designed robust supervisors to enable practical implementation on an experimental testbed. We translated the safety verification problem to an inserted idle-time scheduling problem and approximately solved the problem in polynomially bounded time by adopting a previously studied polynomial-time algorithm solving a scheduling problem with unit process times. In Chapter 5, when modeling an intersection as multiple conflict areas,

we translated the safety verification problem into a jobshop scheduling problem. Because the jobshop scheduling problem is computationally intractable mainly due to its nonconvexity, we approximately solved the problem by formulating two mixed integer linear programming (MILP) problems that yield lower and upper bounds of the optimal cost of the jobshop scheduling problem. In Chapter 6, when considering general and realistic intersection scenarios, with multiple conflict areas and merging/splitting paths, we performed the verification in a hierarchical structure. In the top layer, we employed an abstraction of the actual system and formulated a scheduling problem based on path discretization as an MILP problem. We provided a map between the solutions to the scheduling problem and to the safety verification problem, and thus, could approximately solve the safety verification problem in the low layer.

We now discuss some insights and possible extensions of the works presented in the thesis. Also, we provide some other applications that can be pursued based on the theory developed in the thesis.

7.1 Discussion

As taking more realistic and busier intersections into consideration, we developed the designs of the supervisors. In this section, we discuss how the designs have changed and how they can further be extended for even more realistic scenarios.

We assumed in Chapters 4 and 5 that there is only one vehicle per lane. Our approaches can be easily modified to handle multiple vehicles per lane; one possible modification can be solving the scheduling problems only for the first vehicles in lanes while letting the following vehicles maintain a safe distance from their lead vehicles. Instead of this approach, we presented a less conservative approach based on discretized paths in Chapter 6. Another approach to handling rear-end collisions without any approximation at a single conflict area is provided in [17].

In Chapter 5, the cost error $s^* - s_L^*$ cannot be made as small as desired, where s^* is the maximum tardiness in the jobshop scheduling problem (Problem 5.1) and s_L^* is the lower bound. This is mainly because that the behaviors of the first-order dynamics

allow instantaneous speed changes, thereby not fully representing the behaviors of the second-order dynamics. Thus, we introduced in Problem 6.1 the abstraction (6.5) that includes an additional constraint limiting the instantaneous speed changes of the first-order dynamics. By using this abstraction, we can improve the cost error $s^* - s_L^*$.

Similarly, in Chapter 5, the cost error $s_U^* - s^*$ cannot be made as small as desired, where s_U^* is the upper bound. This is mainly because the upper bound problem (Problem 5.3) accepts an input signal that is maximum inside an intersection. Various upper bound formulations are possible, for example, by allowing only the minimum input inside an intersection. To obtain a smaller cost error, we can formulate another upper bound problem that allows combinations of the maximum and minimum inputs inside an intersection with a binary variable associated with each combination. This approach is less conservative and gives a tighter upper bound, at the expense of computational complexity due to the additional binary variables.

Also, the consideration of uncertainty sources in the supervisor designs is necessary for real-world implementation. Using a similar method used in Chapter 4 allows the extensions of the results in Chapters 5 and 6 to handle the process and measurement errors and the presence of uncontrolled vehicles.

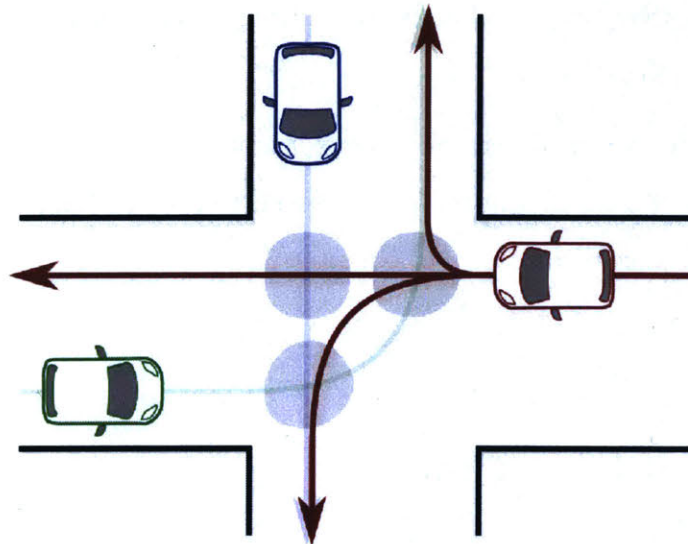


Figure 7-1: Paths of vehicles are undetermined before they enter the intersection.

In the thesis, we assumed that the paths of vehicles are known before they enter an

intersection. One straightforward approach to relaxing this assumption is to prevent collisions on all possible future paths, considering, for example, all three paths in Figure 7-1. A less conservative approach can be incorporating an algorithm that infers future paths of vehicles (e.g., [40]), which remains as future work.

Another interesting extension is to consider moving bad sets. For example, in lane changing scenarios, the location where two paths of vehicles start to merge changes depending on the speed of the vehicles and their steering inputs. One possible approach can be deriving the dynamics of bad sets and using the same scheduling approach presented in this thesis based on the relative dynamics of vehicle dynamics to the bad set dynamics. We expect that this extension will be able to widen application areas of our scheduling approach.

Another possible extension of the works in this thesis is to design a decentralized controller as a scalable and practical solution at intersections where vehicle-to-infrastructure communication technology is not feasible. With the application of air traffic management, decentralized control has been studied in [36,46,62] while it usually terminates with suboptimal solutions or deadlock. At a small and simple intersection, a decentralized approach in the automated management framework was presented in [47]. Designing an effective decentralized supervisor at large intersections remains as future work.

We have published [5] based on the experimental result in Chapter 4, and [3,4] based on the results in Chapter 5. We are currently working on the publication processes of the theoretical results in Chapter 6.

7.2 Future Directions

The thesis provides novel methods to design a decision-making process based on continuous dynamics and coordinate multiple vehicles. These areas have plenty of applications, including automation of city driving, warehouse, and factory.

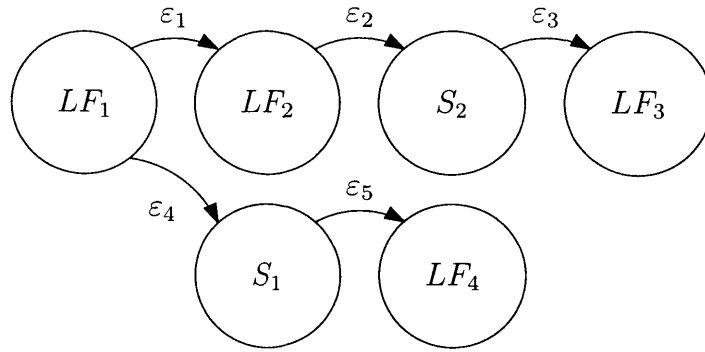


Figure 7-2: Sequence of discrete modes in city driving scenarios. LF_i is the lane following mode in lane i and S_i is the stop mode in lane i .

Decision-Making in City Driving

In scenarios of city driving, driver assistance systems should determine a sequence of decisions, such as whether the ego vehicle has to change lanes at the current time. Figure 7-2 depicts an example of a sequence of modes in city driving scenarios, where LF_i refers to the lane following mode in lane i , and S_i refers to the stop mode in lane i . Here, the problem is to design a controller that determines a decision ε_i such that, for example, it applies ε_1 if the current mode is LF_1 and changing lanes from 1 to 2 is feasible, and ε_2 if the current mode is LF_2 and the vehicle should start braking to reach a full stop before a stop line. The solution to the problem must consider continuous dynamics of vehicles to determine the timing of a discrete input. A similar approach used in this thesis, such as checking whether the current dynamical state is in the backward reachable sets of target sets, can be adopted to solve the problem. An initial result will be presented at 2018 American control conference.

Interactive Environment

An autonomous driving system is designed usually based on an assumption that its decisions do not affect the behaviors of other agents (e.g., pedestrians or other vehicles). In real-world, however, the system interacts with the environment; other agents may not behave as predicted depending on the decisions of the system. This discrepancy can be reduced if vehicle-to-vehicle communications are available with which vehicles can cooperate during the decision-making processes. If there is no

communication available, the autonomous driving system can take a gentle probing to observe other agents' reactions and estimate the resulting future behaviors of other agents (e.g. [51]). The design of an autonomous driving system that interacts with its environment with or without communications would be an interesting future direction.

Factory Automation

The design of controllers for manipulators requires to handle both high-order non-linear dynamics and discrete modes (e.g., move items to another position, assemble items, etc.). In particular, when manipulators collaborate with human operators, one possible approach to their coordination is to make decisions based on the reachable sets of the manipulators to optimize productivity while ensuring no conflict with the human operators. Also, coordinating a large number of mobile robots that autonomously move around warehouses to pick up, pack, and ship specific products can be another future direction.

Appendix A

Proofs in Chapter 4

Theorem 4.1 Problem 4.1 is equivalent to Problem 4.2.

Proof. Note that an instance I of Problem 4.2 is described by $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, and it is the same as an instance of Problem 4.1. Thus, transforming an instance to another takes constant time. Now, the following relation is left to prove the equivalence.

Problem 4.1 accepts $I \Leftrightarrow$ Problem 4.2 accepts I .

(\Rightarrow) Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, there is an input signal $\tilde{\mathbf{u}}_c \in \mathcal{U}_c$ such that $\mathbf{x}(t, \tilde{\mathbf{u}}, \mathbf{d}, \mathbf{s}(\tau)) \notin B$ for all t for all $\tilde{\mathbf{u}}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, and $\mathbf{s}(\tau) \in [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$.

A real number \tilde{T}_i denotes the soonest time at which controlled vehicle i can reach α_i by applying an input signal \tilde{u}_i , that is,

$$x_i(\tilde{T}_i, \tilde{u}_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i.$$

If $x_{h,i}(\tau) \geq \alpha_i$, set $\tilde{T}_i = 0$. Also, a real number \tilde{P}_i denotes the latest time at which vehicle i can reach β_i with an input signal \tilde{u}_i , that is,

$$x_i(\tilde{P}_i, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i.$$

Set $\tilde{P}_i = 0$ if $x_{l,i}(\tau) \geq \beta_i$. Since \tilde{T}_i satisfies the constraint of $P_i(\tilde{T}_i)$ in Definition 4.1,

$$P_i(\tilde{T}_i) \leq \tilde{P}_i.$$

A vector of times $\tilde{\mathbf{T}} = (\tilde{T}_1, \dots, \tilde{T}_n)$ satisfies 4.7 in Problem 4.2. First, \tilde{T}_i is bounded by R_i and D_i by definition. Next, without loss of generality, suppose vehicle i crosses the intersection earlier than vehicle j , that is, $\tilde{T}_i \leq \tilde{T}_j$. Since \tilde{u}_i and \tilde{u}_j guarantee that at most one vehicle is inside an intersection for any uncertainty, we must have $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) \geq \beta_i$ when $x_j(t, \tilde{u}_j, \mathbf{d}_{j,\max}, \mathbf{s}_{h,j}(\tau)) = \alpha_j$. Because x_i is nondecreasing in time, we have $\tilde{P}_i \leq \tilde{T}_j$ and thus $P_i(\tilde{T}_i) \leq \tilde{T}_j$. This concludes that $(\tilde{T}_i, P_i(\tilde{T}_i)) \cap (\tilde{T}_j, P_j(\tilde{T}_j)) = \emptyset$. Lastly, by definition of $\tilde{\mathbf{u}}$, controlled vehicle i and uncontrolled vehicle γ are not simultaneously inside the intersection for all $\tilde{u}_\gamma, \mathbf{d}_\gamma$, and $\mathbf{s}_\gamma(\tau)$. During $(\tilde{T}_i, \tilde{P}_i)$, vehicle i is inside the intersection, and during $(\bar{R}_\gamma, \bar{P}_\gamma)$, vehicle γ is inside the intersection for any uncertainty according to Definition 4.1. Thus, $(\tilde{T}_i, \tilde{P}_i) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. Since $P_i(\tilde{T}_i) \leq \tilde{P}_i$, this implies $(\tilde{T}_i, P_i(\tilde{T}_i)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$.

(\Leftarrow) Given a state estimate $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$, there exists a schedule $\tilde{\mathbf{T}} = (\tilde{T}_1, \dots, \tilde{T}_n) \in \mathbb{R}^n$ that satisfies (4.7) in Problem 4.2.

Let a function \tilde{u}_i satisfy

$$x_i(\tilde{T}_i, \tilde{u}_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i$$

and

$$x_i(P_i(\tilde{T}_i), \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i.$$

If $x_{l,i}(\tau) \geq \beta_i$, we do not consider vehicle i because it has already crossed the intersection. Since x_i depends continuously on \tilde{u}_i , and \mathcal{U}_i is connected by assumption, there exists an input signal \tilde{u}_i in \mathcal{U}_i .

The schedule $\tilde{T}_i \leq \tilde{T}_j$ for controlled vehicles i and j satisfies $P_i(\tilde{T}_i) \leq \tilde{T}_j$, which indicates that $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) \geq \beta_i$ when $x_j(t, \tilde{u}_j, \mathbf{d}_{j,\max}, \mathbf{s}_{h,j}(\tau)) = \alpha_j$. Vehicle j exits the intersection no later than t for all uncertainties, and vehicle j enters the intersection no earlier than t for all uncertainties. With uncontrolled vehicle γ , the schedule \tilde{T}_i satisfies either $\bar{P}_\gamma \leq \tilde{T}_i$ or $P_i(\tilde{T}_i) \leq \bar{R}_\gamma$. In the first case, $x_\gamma(t) \geq \beta_\gamma$ when $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i$, which implies by monotonicity that for all uncertainties, vehicle i enters the intersection after vehicle γ exits. In the second case, $x_\gamma(t) \leq \alpha_\gamma$

when $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \beta_i$, which implies by monotonicity that for all uncertainties, vehicle i exits the intersection before vehicle γ enters. Therefore, an input signal $\tilde{\mathbf{u}}$ satisfies (4.5) in Problem 4.1. \square

Theorem 4.2. Procedure SUPERVISOR in Algorithm 4-2 implements the supervisor s designed in (4.6), and it is nonblocking.

Proof. Given $(\mathbf{s}_m(k\tau), \mathbf{u}_d^k)$, procedure SUPERVISOR returns \mathbf{u}_d^k in line 7 if ans_1 is *yes*. The existence of a feasible solution \mathbf{T}_1 to Problem 4.2 implies that there exists an input signal $\mathbf{u}_c \in \mathcal{U}_c$ (stored as \mathbf{u}_s^{k+1} in the algorithm) that satisfies $\mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{s}_0\tau) \notin B$ for all t for all $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, and $\mathbf{s}_0 \in [\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ where $[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$ is determined by (4.4) with input \mathbf{u}_d . Otherwise, it returns \mathbf{u}_s^k restricted to time $[0, \tau)$ in line 12. This structure corresponds to the supervisor s in (4.6).

We prove the nonblocking property by mathematical induction on time step k . For the base case, we assume that $\text{SUPERVISOR}(\mathbf{s}_m(0), \mathbf{u}_d^0) \neq \emptyset$ and a well-defined input signal \mathbf{u}_s^1 exists. We say \mathbf{u}_s^1 is well-defined if there exists a solution \mathbf{T} to Problem 4.2 that defines \mathbf{u}_s^1 through a map $\sigma([\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)], \mathbf{T})$ given in (4.8). Suppose at step $k-1$, we have $\text{SUPERVISOR}(\mathbf{s}_m((k-1)\tau), \mathbf{u}_d^{k-1}) \neq \emptyset$ and \mathbf{u}_s^k is well-defined. That is, there exists \mathbf{u}_s^k that satisfies

$$\forall \mathbf{s}(k\tau) \in [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)], \quad \mathbf{x}(t, (\mathbf{u}_s^k, \mathbf{u}_{uc}), \mathbf{d}, \mathbf{s}(k\tau)) \notin B. \quad (\text{A.1})$$

for all $t \geq 0$, $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$. Here, $(\mathbf{u}_s^k, \mathbf{u}_{uc})$ indicates an input signal in \mathcal{U} consisting of $\mathbf{u}_s^k \in \mathcal{U}_c$ and $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$.

At at step k , we will show that $\text{SUPERVISOR}(\mathbf{s}_m(k\tau), \mathbf{u}_d^k) \neq \emptyset$ no matter what \mathbf{u}_d^k is applied, and \mathbf{u}_s^{k+1} is well-defined. In Algorithm 4-2, $\text{SUPERVISOR}(\mathbf{s}_m(k\tau), \mathbf{u}_d^k)$ returns either \mathbf{u}_d^k in line 7 or \mathbf{u}_s^k restricted to time $[0, \tau)$ in line 12. In either case, the output of the procedure is nonempty. The former case is when $ans_1 = \text{yes}$ and \mathbf{T}_1 exists, and thus, $\mathbf{u}_s^{k+1} = \sigma([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)], \mathbf{T}_1)$ is well-defined. In the latter case, we consider the state estimate $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ predicted with

\mathbf{u}_s^k . Here, we partition \mathbf{u}_s^k into $\mathbf{u}_{s,[0,\tau]}^k$ and $\mathbf{u}_{s,[\tau,\infty)}^k$ such that

$$\begin{cases} \mathbf{u}_{s,[0,\tau]}^k(t) = \mathbf{u}_s^k(t) & \text{for } t \in [0, \tau), \\ \mathbf{u}_{s,[\tau,\infty)}^k(t - \tau) = \mathbf{u}_s^k(t) & \text{for } t \in [\tau, \infty). \end{cases}$$

That is, $\mathbf{u}_{s,[0,\tau]}^k$ is \mathbf{u}_s^k restricted to time $[0, \tau)$ and $\mathbf{u}_{s,[\tau,\infty)}^k$ is \mathbf{u}_s^k restricted to time $[\tau, \infty)$.

Then, we can rewrite (A.1) as

$$\begin{aligned} \forall \mathbf{s}((k+1)\tau) \in \\ [\mathbf{s}(\tau, (\mathbf{u}_{s,[0,\tau]}^k, \mathbf{u}_{uc,\min}), \mathbf{d}_{\min}, \mathbf{s}_l(k\tau)_{pr}), \mathbf{s}(\tau, (\mathbf{u}_{s,[0,\tau]}^k, \mathbf{u}_{uc,\max}), \mathbf{d}_{\max}, \mathbf{s}_h(k\tau)_{pr})], \\ \mathbf{x}(t, (\mathbf{u}_{s,[\tau,\infty)}^k, \mathbf{u}_{uc}), \mathbf{d}, \mathbf{s}((k+1)\tau)) \notin B, \quad (\text{A.2}) \end{aligned}$$

for all $t \geq 0$, $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, where $\mathbf{u}_{uc,\min}$ and $\mathbf{u}_{uc,\max}$ is the minimum and maximum signals in \mathcal{U}_{uc} , respectively. At step k , we obtain the predicted state estimate $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ given $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po}$ and $\mathbf{u}_{s,[0,\tau]}^k$. Since the posterior estimate is a subset of the prior estimate (i.e., $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{po} \subseteq [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]_{pr}$), the state estimate $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ is a subset of the state interval given in (A.2). Thus, (A.2) is satisfied for all $\mathbf{s}((k+1)\tau) \in [\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$. That is, given $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$, the signal $\mathbf{u}_{s,[\tau,\infty)}^k$ is a solution to Problem 4.1 and by Theorem 4.1, a feasible solution \mathbf{T}_2 to Problem 4.2 exists. The signal $\mathbf{u}_s^{k+1} = \sigma([\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)], \mathbf{T}_2)$ is well-defined.

Therefore, the supervisor is nonblocking. \square

Theorem 4.3. If APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) returns *yes*, then EXACT($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) also returns *yes* (i.e., by Theorem 4.1, there is an input signal satisfying (4.5)).

Proof. Procedure APPROXIMATE of Algorithm 4-5 returns *yes* if a candidate sequence π^* yields a feasible solution \mathbf{T} according to procedure COMPUTE SEQUENCE (line 10). Since $\pi^* \in \mathcal{P}$, where \mathcal{P} is the set of all sequences, EXACT($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) also returns *yes*. \square

To prove Theorem 4.4, we prove two lemmas that compare two procedures AP-

PROXIMATE in Algorithm 4-5 and SOLUTION TO PROBLEM 4.3 in Algorithm 4-4.

Lemma A.1. *If APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\mathbf{T}, yes) , and SOLUTION TO PROBLEM 4.3($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = $(\bar{\mathbf{T}}, \text{yes})$, then $T_i \leq \bar{T}_i$ for all $i \in \{1, \dots, n\}$.*

Proof. For $i \notin \mathcal{M}$, $T_i = \bar{T}_i = 0$. The vector of indexes in \mathcal{M} is denoted by $\pi^* = (j_1, \dots, j_{|\mathcal{M}|})$ in the increasing order of \mathbf{T} , that is, $T_{j_1} \leq T_{j_2} \leq \dots \leq T_{j_{|\mathcal{M}|}}$. We will show by induction on i that $T_{j_i} \leq \bar{T}_{j_i}$ for all $j_i \in \mathcal{M}$. For the base case, $T_{j_1} = \max(R_{j_1}, P_{max})$ in procedure SCHEDULING. Since $\bar{\mathbf{T}}$ is a solution to Problem 4.3, it satisfies $R_{j_1} \leq \bar{T}_{j_1}$ and $(0, P_{max}) \cap (\bar{T}_{j_1}, \bar{T}_{j_1} + \theta_{max}) = \emptyset$. Thus, $\max(R_{j_1}, P_{max}) = T_{j_1} \leq \bar{T}_{j_1}$.

Suppose $T_{j_{i-1}} \leq \bar{T}_{j_{i-1}}$. We need to show that $T_{j_i} \leq \bar{T}_{j_i}$. In procedure SCHEDULING, $T_{j_i} = \max(R_{j_i}, P_{j_{i-1}}(T_{j_{i-1}}))$. If $T_{j_i} = R_{j_i}$, we have $T_{j_i} \leq \bar{T}_{j_i}$ because $\bar{T}_{j_i} \geq R_{j_i}$. If $T_{j_i} = P_{j_{i-1}}(T_{j_{i-1}})$, we have $T_{j_i} \leq \bar{T}_{j_{i-1}} + \theta_{max}$ because $T_{j_{i-1}} \leq \bar{T}_{j_{i-1}}$ and $P_{j_{i-1}}(T_{j_{i-1}}) - T_{j_{i-1}} \leq \theta_{max}$ by definition. Since \bar{T}_{j_i} satisfies $\bar{T}_{j_{i-1}} + \theta_{max} \leq \bar{T}_{j_i}$, we have $T_{j_i} \leq \bar{T}_{j_i}$. For uncontrolled vehicle γ , procedure SCHEDULING assigns the schedule so that it increases to $T_{j_i} = \bar{P}_\gamma$ if $(T_{j_i}, P_{j_i}(T_{j_i})) \cap (\bar{R}_\gamma, \bar{P}_\gamma) \neq \emptyset$. If $(\bar{T}_{j_i}, \bar{T}_{j_i} + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) \neq \emptyset$, the solution $\bar{\mathbf{T}}$ to Problem 4.3 must be $\bar{T}_{j_i} \geq \bar{P}_\gamma$. Therefore, $T_{j_i} \leq \bar{T}_{j_i}$. \square

Lemma A.2. *If APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\emptyset, no) , then SOLUTION TO PROBLEM 4.3($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\emptyset, no) .*

Proof. Procedure APPROXIMATE returns *no* if $[\mathbf{x}_l(\tau), \mathbf{x}_h(\tau)] \cap B \neq \emptyset$ or if $T_i > D_i$ for some i . In the former case, procedure SOLUTION TO PROBLEM 4.3 also returns *no*. In the latter case, suppose SOLUTION TO PROBLEM 4.3($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) returns a nonempty solution $\bar{\mathbf{T}}$. However, $\bar{\mathbf{T}}$ cannot be feasible because $\bar{T}_i \geq T_i$ by Lemma A.1 and $T_i > D_i$. Thus, by contradiction, procedure SOLUTION TO PROBLEM 4.3 returns (\emptyset, no) . \square

Theorem 4.4. *If APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\emptyset, no) , then there is no input signal $\mathbf{u}_c \in \mathcal{U}_c$ that guarantees $\mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{s}_0) \notin \hat{B}$ for all $t \geq 0$ for all $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, and $\mathbf{s}_0 \in [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$.*

Proof. By Lemma A.2, APPROXIMATE($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\emptyset, no) means that SOLUTION TO PROBLEM 4.3($[\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$) = (\emptyset, no) . Thus, we will prove that if there is no schedule satisfying (4.10) in Problem 4.3, there is no input signal to avoid the inflated bad set for all uncertainty. For ease of proof, we prove the contrapositive statement. That is, assume that there is an input signal $\tilde{\mathbf{u}}_c \in \mathcal{U}_c$ such that $\mathbf{x}(t, \tilde{\mathbf{u}}, \mathbf{d}, \mathbf{s}_0) \notin \hat{B}$ for all $t, \tilde{\mathbf{u}}_{uc} \in \mathcal{U}_{uc}, \mathbf{d} \in \mathcal{D}$, and $\mathbf{s}_0 \in [\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)]$. The existence of such an input signal implies that there exists a schedule $\tilde{\mathbf{T}}$ satisfying (4.10). This proof is similar to the first part of the proof of Theorem 4.1.

For controlled vehicle i , let \tilde{T}_i denote the time when $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(\tau)) = \alpha_i$ if $x_{h,i}(\tau) < \alpha_i$, and 0 otherwise. Let \tilde{P}_i denote the time when $x_i(t, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) = \hat{\beta}_i$ if $x_{l,i}(\tau) < \hat{\beta}_i$, and 0 otherwise. For two controlled vehicles i and j , since $x_i(t, \tilde{u}_i, d_i, \mathbf{s}_i(\tau))$ and $x_j(t, \tilde{u}_j, d_j, \mathbf{s}_j(\tau))$ avoid entering the inflated bad set for all uncertainty, $(\tilde{T}_i, \tilde{P}_i) \cap (\tilde{T}_j, \tilde{P}_j) = \emptyset$. Since the inflated intersection takes account of the maximum driving distance during θ_{max} , we have $x_i(\tilde{T}_j + \theta_{max}, \tilde{u}_i, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(\tau)) \leq \hat{\beta}_i$. This implies $\tilde{T}_i + \theta_{max} \leq \tilde{P}_i$ because x_i is nondecreasing in t . Thus, $(\tilde{T}_i, \tilde{T}_i + \theta_{max}) \cap (\tilde{T}_j, \tilde{T}_j + \theta_{max}) = \emptyset$. Uncontrolled vehicle γ enters the intersection no earlier than \bar{R}_γ and exits it no later than \bar{P}_γ for any uncertainty, by definition. Since \tilde{u}_i guarantees that vehicles i and γ never meet inside the intersection for all uncertainty, $(\tilde{T}_i, \tilde{P}_i) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$. Thus $(\tilde{T}_j, \tilde{T}_j + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$ because $\tilde{T}_i + \theta_{max} \leq \tilde{P}_i$. \square

Theorem 4.5. Procedure APPROXIMATESUPERVISOR in Algorithm 4-6 is more restrictive than the supervisor s given by (4.6), that is,

$$\text{APPROXIMATESUPERVISOR}(\mathbf{s}_m(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k \implies s(\mathbf{s}_m(k\tau), \mathbf{u}_d^k) = \mathbf{u}_d^k.$$

Moreover, Algorithm 4-6 is nonblocking.

Proof. APPROXIMATESUPERVISOR($\mathbf{s}_m(k\tau), \mathbf{u}_d^k$) = \mathbf{u}_d^k when APPROXIMATE($[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$) returns *yes*. By Theorem 4.3, there is an input signal $\mathbf{u}_c \in \mathcal{U}_c$ satisfying (4.5). Thus, $s(\mathbf{s}_m(k\tau), \mathbf{u}_d^k)$ returns \mathbf{u}_d^k . This proves that APPROXIMATESUPERVISOR is more restrictive than s , thereby ensuring safety.

The nonblocking property is proved by mathematical induction on time step k . For the base case, we assume that $\text{APPROXIMATESUPERVISOR}(\mathbf{s}_m(0), \mathbf{u}_d^0) = \mathbf{u}_d^0 \neq \emptyset$, in which case \mathbf{u}_s^1 is well-defined, that is, there exists a schedule \mathbf{T} that defines \mathbf{u}_s^1 as $\sigma([\mathbf{s}_l(\tau), \mathbf{s}_h(\tau)], \mathbf{T})$. Suppose at step $k - 1$, we have $\text{APPROXIMATESUPERVISOR}(\mathbf{s}_m((k - 1)\tau), \mathbf{u}_d^{k-1}) \neq \emptyset$ and \mathbf{u}_s^k is well-defined. The input signal \mathbf{u}_s^k satisfies $\dot{\mathbf{x}}(t, (\mathbf{u}_s^k, \mathbf{u}_{uc}), \mathbf{d}, \mathbf{s}_0) \notin B$ for all $\mathbf{u}_{uc} \in \mathcal{U}_{uc}$, $\mathbf{d} \in \mathcal{D}$, and $\mathbf{s}_0 \in [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]$. Now, we need to prove that $\text{APPROXIMATESUPERVISOR}(\mathbf{s}_m(k\tau), \mathbf{u}_d^k) \neq \emptyset$, and \mathbf{u}_s^{k+1} is well-defined.

In Algorithm 4-6, if $ans_1 = yes$ in line 5 or $ans_2 = yes$ in line 11, there exists an input signal that makes the state trajectories avoid entering the bad set by Theorem 4.3. Thus, \mathbf{u}_s^{k+1} is well-defined. Also, the output is nonempty because it is either \mathbf{u}_d^k or \mathbf{u}_s^k restricted to time $[0, \tau)$. In the case when $ans_1 = no$ and $ans_2 = no$, we should prove that $ans_3 = yes$ given a sequence π^{k-1} (i.e., a nonempty solution \mathbf{T}_2 exists in line 13). Note that π^{k-1} is the vector of indexes in the increasing order of the nonzero entries of a feasible schedule \mathbf{T}^{k-1} of the previous step. That is, at step $k - 1$, $\text{SCHEDULING}(\pi^{k-1}, [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]) = (\mathbf{T}^{k-1}, yes)$. We will show that the existence of \mathbf{T}^{k-1} implies that of \mathbf{T}_2 .

We decompose \mathbf{u}_s^k into $\mathbf{u}_{s,[0,\tau)}^k$ and $\mathbf{u}_{s,[\tau,\infty)}^k$ such that $\mathbf{u}_{s,[0,\tau)}^k$ is \mathbf{u}_s^k restricted to time $[0, \tau)$ and $\mathbf{u}_{s,[\tau,\infty)}^k$ is \mathbf{u}_s^k restricted to time $[\tau, \infty)$. The i -th entry of \mathbf{T}^{k-1} is

$$\begin{aligned}
T_i^{k-1} &= \{t : x_i(t, u_{s,i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(k\tau)) = \alpha_i\}, \\
&= \{t : x_i(t, u_{s,[\tau,\infty),i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_i(\tau, u_{s,[0,\tau),i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(k\tau))) = \alpha_i\} + \tau, \\
&\leq \underbrace{\{t : x_i(t, u_{s,[\tau,\infty),i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_i(\tau, u_{s,[0,\tau),i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(k\tau)_{po})) = \alpha_i\}}_{:=\tilde{T}_i} + \tau.
\end{aligned} \tag{A.3}$$

The inequality is due to $[\mathbf{s}_{l,i}(k\tau), \mathbf{s}_{h,i}(k\tau)]_{po} \subseteq [\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]$ by (4.3) and the mono-

tonicity of the dynamics. Similarly, the process time $P_i(T_j^{k-1})$ is

$$\begin{aligned}
P_i(T_i^{k-1}) &= \{t : x_i(t, u_{s,i}^k, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(k\tau)) = \beta_i\}, \\
&= \{t : x_i(t, u_{s,[\tau,\infty),i}^k, \mathbf{d}_{i,\min}, \mathbf{s}_i(\tau, u_{s,[0,\tau),i}^k, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(k\tau)) = \beta_i\} + \tau, \\
&\geq \underbrace{\{t : x_i(t, u_{s,[\tau,\infty),i}^k, \mathbf{d}_{i,\min}, \mathbf{s}_i(\tau, u_{s,[0,\tau),i}^k, \mathbf{d}_{i,\min}, \mathbf{s}_{l,i}(k\tau)_{po}) = \beta_i\}}_{:=\tilde{P}_i} + \tau.
\end{aligned} \tag{A.4}$$

Then, we will show that the schedule $\tilde{\mathbf{T}} = (\tilde{T}_1, \dots, \tilde{T}_{n_c})$ is the solution returned by SCHEDULING($\pi^{k-1}, [\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$). The release time and deadline of the procedure is by definition,

$$\begin{aligned}
R_i &= \min_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(k+1)\tau) = \alpha_i\}, \\
D_i &= \max_{u_i \in \mathcal{U}_i} \{t : x_i(t, u_i, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}((k+1)\tau) = \alpha_i\}.
\end{aligned}$$

Because $\mathbf{s}_i(\tau, u_{s,[0,\tau),i}^k, \mathbf{d}_{i,\max}, \mathbf{s}_{h,i}(k\tau)_{po}) = \mathbf{s}_h((k+1)\tau)$ by definition given in (4.4), we have $\tilde{T}_i \in [R_i, D_i]$. Since \mathbf{T}^{k-1} is feasible, $(T_i^{k-1}, P_i(T_i^{k-1})) \cap (T_j^{k-1}, P_j(T_j^{k-1})) = \emptyset$ for $i \neq j \in \{1, 2, \dots, n\}$. Because of (A.3) and (A.4),

$$(\tilde{T}_i, \tilde{P}_i) \subseteq (T_i^{k-1}, P_i^{k-1}(T_i^{k-1})) - \tau.$$

Since $P_i(\tilde{T}_i) \leq \tilde{P}_i$, we have $(\tilde{T}_i, P_i(\tilde{T}_i)) \cap (\tilde{T}_j, P_j(\tilde{T}_j)) = \emptyset$. For uncontrolled vehicle γ , an idle-time given $[\mathbf{s}_l(k\tau), \mathbf{s}_h(k\tau)]$ is denoted by $(\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1})$, and an idle-time given $[\mathbf{s}_l((k+1)\tau), \mathbf{s}_h((k+1)\tau)]$ is denoted by $(\bar{R}_\gamma, \bar{P}_\gamma)$. We can easily show that

$$(\bar{R}_\gamma, \bar{P}_\gamma) \subseteq (\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1}) - \tau.$$

. Thus, $(T_i^{k-1}, P_i(T_i^{k-1})) \cap (\bar{R}_\gamma^{k-1}, \bar{P}_\gamma^{k-1}) = \emptyset$ implies that $(\tilde{T}_i, P_i(\tilde{T}_i)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$.

Thus, $\tilde{\mathbf{T}}$ is feasible, thereby implying $ans_3 = yes$ in line 13 of Algorithm 4-6. Since $\mathbf{T}_2 = \tilde{\mathbf{T}}$ exists, $\mathbf{k} + 1, \infty$ is well-defined.

Therefore, the supervisor is nonblocking. \square

Appendix B

Proofs in Chapter 5

Theorem 5.1. Problem 2.1 is equivalent to Problem 5.1.

Proof. An instance I of Problem 2.1 is described by $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$. Since an instance of Problem 5.1 is identical to I , the transformation between instances takes $O(1)$ time. Thus, all we have to show is that

$$\text{Problem 2.1 accepts } I \Leftrightarrow \text{Problem 5.1 accepts } I,$$

where we say Problem 5.1 accepts I if $s^* = 0$.

(\Rightarrow) Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, there exists an input signal $\tilde{\mathbf{u}} \in \mathcal{U}$ such that $\mathbf{x}(t, \tilde{\mathbf{u}}) \notin B$ for all $t \geq 0$. In this proof, we assume $x_j(0) < \alpha_{i,j}$. For all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$, let

$$\tilde{T}_{i,j} = \{t : x_j(t, \tilde{u}_j) = \alpha_{i,j}\}, \quad \tilde{P}_{i,j} = \{t : x_j(t, \tilde{u}_j) = \beta_{i,j}\}.$$

We will show that $\tilde{\mathbf{T}} = \{\tilde{T}_{i,j} : (i, j) \in \mathcal{N}_{\mathbf{x}(0)}\}$ satisfies the constraints in Problem 5.1 and gives $s^* = 0$. By the definitions of $R_{i,j}$ and $D_{i,j}$, we have $R_{i,j}(\tilde{\mathbf{T}}) \leq \tilde{T}_{i,j} \leq D_{i,j}(\tilde{\mathbf{T}})$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ and thus $s^* = 0$. For all $(i, j) \leftrightarrow (i, j') \in \mathcal{D}$, assume without loss of generality vehicle j enters conflict area i before vehicle j' . Then we know that at $t = \tilde{P}_{i,j}$, since $x_j(t, \tilde{u}_j) = \beta_{i,j}$, we have $x_{j'}(t, \tilde{u}_j) \leq \alpha_{i,j'}$. That is, $\tilde{P}_{i,j} \leq \tilde{T}_{i,j'}$ due to the monotonicity. The constraints given in the definitions of $P_{i,j}$ are satisfied with

the input signal \tilde{u}_j , thereby implying $P_{i,j} \leq \tilde{P}_{i,j}$. Therefore, $P_{i,j} \leq \tilde{T}_{i,j'}$.

(\Leftarrow) Given a state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, there exists a schedule $\hat{\mathbf{T}}$ satisfying the constraints in Problem 5.1 and yielding $s^* = 0$. Because $\hat{T}_{i,j} \in [R_{i,j}(\hat{\mathbf{T}}), D_{i,j}(\hat{\mathbf{T}})]$ and \mathcal{U}_j is connected, there exists $\hat{u}_j \in \mathcal{U}_j$ that satisfies $x_j(\hat{T}_{i,j}, \hat{u}_j) = \alpha_{i,j}$ and $x_j(P_{i,j}, \hat{u}_j) = \beta_{i,j}$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$. For all $(i, j) \leftrightarrow (i, j') \in \mathcal{D}$, we have $P_{i,j}(\hat{\mathbf{T}}) \leq \hat{T}_{i,j'}$ if $\hat{T}_{i,j} \leq \hat{T}_{i,j'}$. Then, at $t = P_{i,j'}(\hat{\mathbf{T}})$, we have $x_j(t, \hat{u}_j) = \beta_{i,j}$ while $x_{j'}(t, \hat{u}_{j'}) \leq \alpha_{i,j'}$. This implies that any two vehicles never meet inside a conflict area, that is, $\mathbf{x}(t, \hat{\mathbf{u}}) \notin B$ for all $t \geq 0$. \square

Theorem 5.2. $s_L^* \leq s^*$

Proof. Suppose Problem 5.1 finds \mathbf{T}^* and \mathbf{k}^* with the corresponding cost s^* , whether or not $s^* = 0$. We will show that $\tilde{\mathbf{t}} = \mathbf{T}^*$ and $\tilde{\mathbf{k}} = \mathbf{k}^*$ become a feasible solution to Problem 5.2 with some $\tilde{\mathbf{p}}$.

Given \mathbf{T}^* , we have $P_{i,j}(\mathbf{T}^*)$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$ according to the definitions in (5.3) and (5.4). Consider $\tilde{p}_{i,j} = P_{i,j}(\mathbf{T}^*)$. Then $P_{i,j}(\mathbf{T}^*) - T_{i,j}^* = \tilde{p}_{i,j} - \tilde{t}_{i,j}$ is the time to reach $\beta_{i,j}$ from $\alpha_{i,j}$ and thus satisfies (P5.2.2). Constraint (P5.2.3) is the same as constraint (P5.1.2) in Problem 5.1.

We will show that $r_{i,j}(\tilde{\mathbf{p}}) \leq R_{i,j}(\mathbf{T}^*)$ and $D_{i,j}(\mathbf{T}^*) \leq d_{i,j}(\tilde{\mathbf{p}})$. As mentioned earlier, for $(i, j) \in \mathcal{F}$, $r_{i,j} = R_{i,j}$ and $d_{i,j} = D_{i,j}$. For $(i', j) \rightarrow (i, j) \in \mathcal{C}$, $R_{i,j}$ is equal to $T_{i',j}^*$ plus the minimum time to reach $\alpha_{i,j}$ from $\alpha_{i',j}$ by (5.1). Considering this definition with (5.3) and (5.4), $R_{i,j}$ is again equal to $P_{i',j}$ plus the minimum time to reach $\alpha_{i,j}$ from $\beta_{i',j}$, thereby $R_{i,j} \geq P_{i',j} + (\alpha_{i,j} - \beta_{i',j})/v_{j,\max}$. Also, since $P_{i',j} = \tilde{p}_{i',j}$, we have $P_{i',j} + (\alpha_{i,j} - \beta_{i',j})/v_{j,\max} = r_{i,j}$. Thus $r_{i,j} \leq R_{i,j}$. Similarly, $D_{i,j} \leq d_{i,j}$. By these inequalities, $R_{i,j} \leq T_{i,j}^*$ implies $r_{i,j} \leq T_{i,j}^* = \tilde{t}_{i,j}$, which is constraint (P5.2.1).

Therefore, $\tilde{\mathbf{t}}$, $\tilde{\mathbf{p}}$, and $\tilde{\mathbf{k}}$ is a feasible solution of Problem 5.2. Since $D_{i,j} \leq d_{i,j}$, $s_L^* \leq \max(\tilde{t}_{i,j} - d_{i,j}, 0) \leq \max(T_{i,j}^* - D_{i,j}, 0) = s^*$. \square

Theorem 5.3. $s_U^* = 0 \Rightarrow s^* = 0$.

Proof. Suppose Problem 5.3 finds an optimal solution $\mathbf{T}^{\mathcal{F}*} = \{T_{i,j}^{\mathcal{F}*} : (i, j) \in \mathcal{F}\}$ and $\mathbf{k}^* = \{k_{ijj'}^* : (j, j') \in \mathcal{D}_i\}$ that yields $s_U^* = 0$ given an initial condition $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$. Because $T_{i,j}^{\mathcal{F}*} \in [\bar{R}_{i,j}, \bar{D}_{i,j}]$, there exists an input signal $u_j^*(\cdot) \in \mathcal{U}_j$ such that $x_j(T_{i,j}^{\mathcal{F}*}, u_j^*(\cdot)) =$

$\alpha_{i,j}$. We define $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{P}}$ as follows: for $(i, j) \in \mathcal{F}$, if $x_j(0) < \alpha_{i,j}$,

$$\begin{aligned}\tilde{T}_{i,j} &= T_{i,j}^{\mathcal{F}*}, \\ \tilde{P}_{i,j} &= t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}; x_j(T_{i,j}^{\mathcal{F}*}, u_j(\cdot)) - \alpha_{i,j}),\end{aligned}$$

If $x_j(0) \geq \alpha_{i,j}$, let $\tilde{T}_{i,j} = 0$ and $\tilde{P}_{i,j} = t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j})$.

For $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$ with the first operation $(i', j) \in \mathcal{F}$,

$$\begin{aligned}\tilde{T}_{i,j} &= t_{\min}(x_j(0), \dot{x}_j(0), \alpha_{i,j}; x_j(T_{i',j}^{\mathcal{F}*}, u_j(\cdot)) - \alpha_{i',j}), \\ \tilde{P}_{i,j} &= t_{\min}(x_j(0), \dot{x}_j(0), \beta_{i,j}; x_j(T_{i',j}^{\mathcal{F}*}, u_j(\cdot)) - \alpha_{i',j}).\end{aligned}$$

Note that $\tilde{P}_{i,j}$ is $P_{i,j}(\tilde{\mathbf{T}})$ by the definitions in (5.3) and (5.4). We will show that $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{k}} = \mathbf{k}^*$ is a feasible solution in Problem 5.1 that yields $s^* = 0$.

For constraint (P5.1.1) in Problem 5.1, since $T_{i,j}^{\mathcal{F}*}$ satisfies constraint (P5.3.1) and $\bar{R}_{i,j} = R_{i,j}$ for all $(i, j) \in \mathcal{F}$, we have $R_{i,j} \leq \tilde{T}_{i,j}$. For $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$, we define $\tilde{T}_{i,j}$ as the minimum time to reach conflict area i , thereby $\tilde{T}_{i,j} = R_{i,j}(\tilde{\mathbf{T}})$ by (5.1). This establishes that $R_{i,j} \leq \tilde{T}_{i,j}$ for all $(i, j) \in \mathcal{N}_{\mathbf{x}(0)}$. For constraint (P5.1.2) in Problem 5.1, let us compare $\tilde{T}_{i,j}$ and $\tilde{P}_{i,j}$ with $\bar{T}_{i,j}(\mathbf{T}^{\mathcal{F}*})$ and $\bar{P}_{i,j}(\mathbf{T}^{\mathcal{F}*})$ in Definition 5.2. If $x_j(0) < \alpha_{j,\min}$, we have $\bar{T}_{i,j} \leq \tilde{T}_{i,j}$ and $\bar{P}_{i,j} \leq \tilde{P}_{i,j}$. If $x_j(0) \geq \alpha_{j,\min}$, we have $\bar{T}_{i,j} = \tilde{T}_{i,j}$ and $\bar{P}_{i,j} = \tilde{P}_{i,j}$ because $s_{\mathcal{U}}^* = 0$ implies $T_{i',j}^{\mathcal{F}*} = \bar{R}_{i',j}$ for $(i', j) \in \mathcal{F}$. Thus, constraint (P5.3.2) becomes

$$\begin{aligned}\tilde{P}_{i,j} &\leq \bar{P}_{i,j} \leq \bar{T}_{i,j'} + M(1 - k_{i,j'}^*) \leq \tilde{T}_{i,j'} + M(1 - k_{i,j'}^*), \\ \tilde{P}_{i,j'} &\leq \bar{P}_{i,j'} \leq \bar{T}_{i,j} + M(1 - k_{i,j}^*) \leq \tilde{T}_{i,j} + M(1 - k_{i,j}^*).\end{aligned}$$

That is, $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{k}} = \mathbf{k}^*$ satisfy constraint (P5.1.2).

Now we have a feasible solution $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{k}}$ in Problem 5.1. For $(i, j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}$, we have $\tilde{T}_{i,j} = R_{i,j} \leq D_{i,j}$, and thus, $\max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)} \setminus \mathcal{F}} (\tilde{T}_{i,j} - D_{i,j}, 0) = 0$. For $(i, j) \in \mathcal{F}$,

we have $\bar{D}_{i,j} \leq D_{i,j}$ and the following inequalities complete the proof.

$$\begin{aligned} s^* &\leq \max_{(i,j) \in \mathcal{N}_{\mathbf{x}(0)}} (\tilde{T}_{i,j} - D_{i,j}, 0) = \max_{(i,j) \in \mathcal{F}} (\tilde{T}_{i,j} - D_{i,j}, 0) \\ &\leq \max_{(i,j) \in \mathcal{F}} (\tilde{T}_{i,j} - \bar{D}_{i,j}, 0) = s_U^*. \end{aligned}$$

Therefore, if $s_U^* = 0$, we have $s^* = 0$. □

Theorem 5.6. Algorithm 5-7 guarantees that the system's state never enters the bad set, that is, $\mathbf{x}(t) \notin B$ for all $t \geq 0$, and is nonblocking.

Proof. By construction, the supervisor guarantees that the state never enters the bad set. The supervisor allows the desired input signal if $s_U^* = 0$ given the desired state. This implies by Theorems 5.1 and 5.3 that there exists an input signal such that a state trajectory starting from the desired state avoids the bad set. If $s_U^* > 0$ given the desired state, the supervisor takes temporary control of the drivers at the current time to drive the system state to the state from which $s_U^* = 0$.

We prove the nonblocking property of the supervisor by induction on time step k . For the base case, assume $\text{SUPERVISOR}(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{u}_d^0) \neq \emptyset$ and \mathbf{u}_s^1 is well-defined, i.e., there is a schedule to Problem 5.3 corresponding to $s_U^* = 0$ that defines $\mathbf{u}_{safe}^{1,\infty}$ using the map σ . Suppose at time $(k-1)\tau$,

$$\text{SUPERVISOR}(\mathbf{x}((k-1)\tau), \dot{\mathbf{x}}((k-1)\tau), \mathbf{u}_d^{k-1}) \neq \emptyset,$$

and \mathbf{u}_s^k is well-defined. At time $k\tau$, for any $\mathbf{u}_d^k \in \mathbf{U}$,

$$\text{SUPERVISOR}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{u}_d^k) \neq \emptyset$$

because it returns either \mathbf{u}_d^k or \mathbf{u}_s^k restricted to time $[0, \tau)$. We need to show that \mathbf{u}_s^{k+1} is well-defined in line 6 or line 12 of Algorithm 5-7. In line 6, since $\mathbf{T}_1^{\mathcal{F}*} = \{T_{i,j}^{\mathcal{F}*} : (i,j) \in \mathcal{F}\}$ yields $s_U^* = 0$, we have $\bar{R}_{i,j} \leq T_{i,j}^{\mathcal{F}*} \leq \bar{D}_{i,j}$ for all $(i,j) \in \mathcal{F}$, and thus, there exists $u_j \in \mathcal{U}_j$ in the connected set \mathcal{U}_j that satisfies $x_j(T_{i,j}^{\mathcal{F}*}, u_j, x_{d,j}^k, \dot{x}_{d,j}^k) = \alpha_{i,j}$. Thus, $\sigma(\mathbf{x}_d^k, \dot{\mathbf{x}}_d^k, \mathbf{T}_1^{\mathcal{F}*})$ is well-defined. In the case of line 12, we have that $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ is

either $(\mathbf{x}_d^{k-1}, \dot{\mathbf{x}}_d^{k-1})$ or $(\mathbf{x}_s^{k-1}, \dot{\mathbf{x}}_s^{k-1})$ depending on the output of the supervisor at the previous time step. In either case, there exists a safe input signal \mathbf{u}_s^k by the induction hypothesis. Since the safe state $(\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k)$ is the state reached after time τ with the safe input signal \mathbf{u}_s^k , the safe signal \mathbf{u}_s^k restricted to time $[\tau, \infty)$ guarantees that the state starting from the safe state $(\mathbf{x}_s^k, \dot{\mathbf{x}}_s^k)$ can avoid the bad set. By Theorem 5.1, $\mathbf{T}_2^{\mathcal{F}^*}$ yields $s_U^* = 0$ and thus \mathbf{u}_s^{k+1} is well-defined. \square

Bibliography

- [1] H. Ahn. Semi-autonomous control of multiple heterogeneous vehicles for intersection collision avoidance. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, United States, June 2014.
- [2] H. Ahn, A. Colombo, and D. Del Vecchio. Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles. In *Proc. American Control Conf. (ACC)*, pages 867–873, June 2014.
- [3] H. Ahn and D. Del Vecchio. Semi-autonomous intersection collision avoidance through job-shop scheduling. In *Proc. Int. Conf. Hybrid Syst.: Computation and Control (HSCC)*, pages 185–194, April 2016.
- [4] H. Ahn and D. Del Vecchio. Safety verification and control for collision avoidance at road intersections. *IEEE Trans. Autom. Control*, PP(99):1–1, 2017.
- [5] H. Ahn, A. Rizzi, A. Colombo, and D. Del Vecchio. Experimental testing of a semi-autonomous multi-vehicle collision avoidance algorithm at an intersection testbed. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, pages 4834–4839, September 2015.
- [6] F. Althché, X. Qian, and A. de La Fortelle. Least restrictive and minimally deviating supervisor for safe semi-autonomous driving at an intersection: An MIQP approach. In *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, November 2016.
- [7] F. Althché, X. Qian, and A. de La Fortelle. An algorithm for supervised driving of cooperative semi-autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.*, 18(12):3527–3539, Dec 2017.
- [8] R. Alur, T. Dang, and F. Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *ACM Trans. Embedded Computing Syst.*, 5(1):152–199, February 2006.
- [9] L. Alvarez and R. Horowitz. Analysis and verification of the PATH AHS coordination-regulation layers hybrid system. In *Proc. American Control Conf. (ACC)*, pages 2460–2461, June 1997.

- [10] L. Alvarez and R. Horowitz. Hybrid controller design for safe maneuvering in the PATH AHS architecture. In *Proc. American Control Conf. (ACC)*, pages 2454–2459, June 1997.
- [11] D. Angeli and E.D. Sontag. Monotone control systems. *IEEE Trans. Autom. Control*, 48(10):1684–1698, October 2003.
- [12] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, volume 1790, pages 20–31. Springer, March 2000.
- [13] D. Bresch-Pietri and D. Del Vecchio. Estimation for decentralized safety control under communication delay and measurement uncertainty. *Automatica*, 62:292–303, December 2015.
- [14] L. Bruni, A. Colombo, and D. Del Vecchio. Robust multi-agent collision avoidance through scheduling. In *Proc. IEEE Conf. Decision and Control (CDC)*, pages 3944–3950, December 2013.
- [15] A. Colombo and D. Del Vecchio. Supervisory control of differentially flat systems based on abstraction. In *Proc. IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC)*, pages 6134–6139, December 2011.
- [16] A. Colombo and D. Del Vecchio. Efficient algorithms for collision avoidance at intersections. In *Proc. Int. Conf. Hybrid Syst.: Computation and Control (HSCC)*, pages 145–154, April 2012.
- [17] A. Colombo and D. Del Vecchio. Least restrictive supervisors for intersection collision avoidance: A scheduling approach. *IEEE Trans. Autom. Control*, 60(6):1515–1527, June 2015.
- [18] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Courier Dover Publications, 2003.
- [19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, 3rd edition edition, 2009.
- [20] E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune. Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. In *Proc. American Control Conf. (ACC)*, March 2013.
- [21] E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune. Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. *Discrete Event Dynamic Syst.*, 27(1):1–44, March 2017.
- [22] D. Del Vecchio, M. Malisoff, and R. Verma. A separation principle for a class of hybrid automata on a partial order. In *Proc. American Control Conf. (ACC)*, pages 3638–3643, June 2009.

- [23] T. Dreossi, T. Dang, and C. Piazza. Parallelotope bundles for polynomial reachability. In *Proc. Int. Conf. Hybrid Syst.: Comput. and Control (HSCC)*, pages 297–306, April 2016.
- [24] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *J. Artificial Intell. Research*, 31:591–656, March 2008.
- [25] C. A. Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.
- [26] P. Gagarinov and A. A. Kurzhanskiy. *Ellipsoidal Toolbox*, 2014.
- [27] M. R. Garey, D. S. Johnson, B. B. Simons, and R. E. Tarjan. Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. Computing*, 10(2):256–269, May 1981.
- [28] R. Ghaemi and D. Del Vecchio. Control for safety specifications of systems with imperfect information on a partial order. *IEEE Trans. Autom. Control*, 59(4):982–995, April 2014.
- [29] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Syst.*, 18(2):163–179, October 2007.
- [30] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio. Cooperative collision avoidance at intersections: algorithms and experiments. *IEEE Trans. Intell. Transp. Syst.*, 14(3):1162–1175, April 2013.
- [31] M. R. Hafner and D. Del Vecchio. Computational tools for the safety control of a class of piecewise continuous systems with imperfect information on a partial order. *SIAM J. Control and Optimization*, 49(6):2463–2493, December 2011.
- [32] T. A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. Autom. Control*, 43(4):540–554, April 1998.
- [33] IBM Corporation. *CPLEX User’s Manual*, 2015.
- [34] M. A. S. Kamal, J. Imura, T. Hayakawa, A. Ohata, and K. Aihara. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. *IEEE Trans. Intell. Transp. Syst.*, 16(3), September 2014.
- [35] J. J. Kanet and V. Sridharan. Scheduling with inserted idle time: Problem taxonomy and literature review. *Operations Research*, 48(1):99–110, February 2000.
- [36] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas. Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Trans. Control Syst. Technol.*, 16(1):19–33, January 2008.

- [37] K. Kim and P. R. Kumar. An MPC-based approach to provable system-wide safety and liveness of autonomous ground traffic. *IEEE Trans. Autom. Control*, 59(12):3341–3356, December 2014.
- [38] J. Lee and B. Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Trans. Intell. Transp. Syst.*, 13(1):81–90, January 2012.
- [39] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, March 1999.
- [40] P. Lytrivis, G. Thomaidis, M. Tsogas, and A. Amditis. An advanced cooperative path prediction algorithm for safety applications in vehicular networks. *IEEE Trans. Intell. Transp. Syst.*, 12(3):669–679, September 2011.
- [41] Massachusetts Department of Transportation. 2012 Top Crash Locations Report, 2014.
- [42] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Autom. Control*, 50(7):947–957, July 2005.
- [43] T. Moor and J. Raisch. Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *Modelling, Analysis, and Design of Hybrid Systems*, pages 247–265. Springer, July 2002.
- [44] A. I. Morales Medina, N. van de Wouw, and H. Nijmeijer. Cooperative intersection control based on virtual platooning. *IEEE Trans. Intell. Transp. Syst.*, pages 1–14, August 2017, in press.
- [45] N. Murgovski, G. R. de Campos, and J. Sjöberg. Convex modeling of conflict resolution at traffic intersections. In *Proc. IEEE Conf. Decision and Control (CDC)*, pages 4708–4713, December 2015.
- [46] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli. Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems. *IEEE Trans. Robot.*, 23(6):1170–1183, December 2007.
- [47] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S. Y. Liu, M. Novitzky, I. F. Okuyama, J. Papis, G. Rosman, V. Varricchio, H. C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *Proc. IEEE Int. Conf. Robot. and Autom. (ICRA)*, pages 1497–1504, May 2017.
- [48] J. Peng and S. Akella. Coordinating Multiple Double Integrator Robots on a Roadmap: Convexity and Global Optimality. In *Proc. IEEE Int. Conf. Robot. and Autom. (ICRA)*, pages 2751–2758, April 2005.

- [49] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *Int. J. Robot. Research*, 24(4):295–310, April 2005.
- [50] M. Pinedo. *Scheduling*. Springer, 4th edition, 2012.
- [51] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan. Information gathering actions over human internal state. In *IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, pages 66–73, Oct 2016.
- [52] Thomas B Sheridan. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.
- [53] Barbara Simons. A fast algorithm for single processor scheduling. In *Annu. Symp. Foundations of Computer Science*, pages 246–252, 1978.
- [54] J. E. Slotine and W. Li. Applied nonlinear control. *Prentice-Hall*, 1991.
- [55] P. Stone and M. Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.
- [56] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti. Revisiting street intersections using slot-based systems. *PLoS ONE*, 11(3):e0149607, March 2016.
- [57] C. Tomlin, J. Lygeros, and Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proc. IEEE*, 88(7):949–970, July 2000.
- [58] C. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proc. IEEE*, 91(7):986–1001, July 2003.
- [59] C. Tomlin, George J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Trans. Autom. Control*, 43(4):509–521, April 1998.
- [60] U.S. Department of Transportation. ITS Strategic research plan 2015-2019. <http://www.its.dot.gov/strategicplan.pdf>, 2014.
- [61] U.S. Department of Transportation National Highway Safety Administration. Traffic Safety Facts 2012. <http://www-nrd.nhtsa.dot.gov/Pubs/812032.pdf>, 2012.
- [62] W. Zhang, M. Kamgarpour, D. Sun, and C. J. Tomlin. A hierarchical flight planning framework for air traffic management. *Proc. IEEE*, 100(1):179–194, January 2012.