Semi-autonomous Control of Multiple Heterogeneous Vehicles for Intersection Collision Avoidance

by

Heejin Ahn

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

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

© Massachusetts Institute of Technology 2014. All rights reserved.

Signature redacted

Author

Department of Mechanical Engineering

May 27, 2014

Signature redacted

Certified by

Domitilla Del Vecchio
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by

David E. Hardt
Chairman, Department Committee on Graduate Theses
Semi-autonomous Control of Multiple Heterogeneous Vehicles for Intersection Collision Avoidance

by

Heejin Ahn

Submitted to the Department of Mechanical Engineering
on May 27, 2014, in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Abstract

This paper describes the design of a supervisory controller (supervisor) that manages multiple heterogeneous vehicles, i.e., multiple controlled and uncontrolled vehicles, to avoid intersection collisions. Two main problems are addressed: verification of the safety of all vehicles at an intersection, and management of the inputs of controlled vehicles. For the verification problem, we employ an inserted idle-time scheduling approach, where the "inserted idle-time" is a time interval when the intersection is deliberately held idle for uncontrolled vehicles to safely cross the intersection. For the management problem, we design a supervisor that is least restrictive in the sense that it overrides controlled vehicles only when a safety violation becomes imminent. We analyze computational complexity and propose an efficient version of the supervisor with a quantified approximation bound. To mitigate the abrupt changes of control inputs and to reduce the number of unnecessary interventions, we additionally design two optimization problems and provide the supervisor with a more conservative bound.

Thesis Supervisor: Domitilla Del Vecchio
Title: Associate Professor
Acknowledgments

First and foremost, I would like to express a sincere appreciation to my thesis supervisor, Prof. Domitilla Del Vecchio. I am particularly thankful for her thoroughness in discussing research directions and editing my naive papers. I always appreciate her interest in my research problems and her effort to provide anything that can help me tackle the problems. Her uncountable advice and encouragement have been a constant driving force behind my research work.

I benefited from interaction with the past and current members of Prof. Del Vecchio's group for their valuable comments and friendship. In particular, I wish to acknowledge the help provided by Dr. Alessandro Colombo, who is now an assistant professor of Politecnico di Milano. His suggestions have inspired me to analyze problems correctly and thoroughly. I would like to thank Dr. Daniel Hoehener for his help on formulating optimization problems. Without his experience, I could not complete the optimal approaches in this paper. I wish to thank Andrea for helping me run experiments. Because of my initial ignorance of Linux and HTTP, it might be tough for him to teach everything to me.

I am grateful to the members of the MIT Korean Graduate Student Association of Mechanical Engineering (KGSAME) for making my life at MIT much more pleasant and exciting. I would also like to express my gratitude to my friends who take classes and study together, to my friends from First Korean Church in Cambridge for praying for me, and to my friends in Korea for always being emotionally close to me. My special thanks go to Minkyun, who encourages me and makes me always able to count on him. Thank you for being here with me.

I owe everything to my family, particularly to my parents, Kookyoung Ahn and Kyoungsoo Hong, for having given me unconditional love and support. I am also thankful to my sister and brother, Heeyoon and Junhong, and to my grandmother for their constant care. Without them, this thesis could not be possible.
## Contents

1 Introduction ........................................... 15

2 Basic Concepts ..................................... 19
   2.1 Mathematical Background ............................ 19
       2.1.1 Scheduling .................................. 19
       2.1.2 Equivalence ................................ 21
   2.2 Intersection Model ................................. 21
   2.3 Dynamic Model .................................. 22

3 An Exact Approach ................................. 25
   3.1 Problem Statement ................................ 25
       3.1.1 Verification Problem .......................... 25
       3.1.2 Supervisory Problem ...................... 29
   3.2 Problem Solutions ................................ 30
       3.2.1 Solution of the Verification Problem .......... 30
       3.2.2 Solution of the Supervisory Problem .......... 32
   3.3 Simulation Results .............................. 36

4 An Efficient Approach .......................... 39
   4.1 Relaxed Verification Problem .................... 39
       4.1.1 Relaxed Scheduling Problem .............. 39
       4.1.2 Relaxed Verification Problem ............ 42
   4.2 Efficient Supervisors ........................... 44
List of Figures

2-1 An example of 3 vehicles approaching an intersection. The traffic intersection (left) is modeled as one point at which unidirectional paths intersect (right). Here, $\alpha_i$ and $\beta_i$ represent the location of the intersection on the longitudinal path $i$. ......................... 21

3-1 Positions of 5 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by the supervisor $s$ in Section 3.2 (Algorithm 3.2). The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The shaded region represents the location of the intersection. ...................... 37

3-2 Positions of 8 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by the supervisor $s$ in Section 3.2 (Algorithm 3.2). ......................... 38

4-1 Positions of 8 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by $s_{approx}$ in Section 4.2. The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The darker shaded region represents the location of the intersection while the lighter shaded region represents that of the inflated intersection. ......................... 45
5-1 Positions of 4 vehicles (black solid line). The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The darker shaded region represents the location of the intersection while the lighter shaded region represents that of the inflated intersection.

5-2 Control input (black solid line) for the last car with its desired input (blue dotted line). The two horizontal dotted lines represent the upper and lower bounds for the input.

5-3 Positions of 4 vehicles where the last vehicle is far from the intersection compared to the others.

5-4 Control input (black solid line) for the last car in Figure 5-3(a) with its desired input (blue dotted line). The two horizontal dotted lines represent the upper and lower bounds for the input.

5-5 Average cost on 60 samples of four vehicles.

5-6 Implementation of the two optimization problems (Problem 5.3 and Problem 5.4) at stage $k$. First, the Discrete Scheduling Problem is solved via Algorithm 4.4, and if there exists a feasible schedule, the sequence of the schedule lets the Schedule Optimization Problem compute the optimal schedule $T^*$. If there does not exist a feasible schedule, Algorithm 4.4 returns no and an empty schedule. Depending on this answer, the supervisor either runs the Input Optimization Problem to store the safe input for the next step $u_{k+1, safe}$ or overrides the system using the safe input $u_{k, safe}$.
5-7 The same initial condition and desired input are considered as in Figure 5-3. In Figure 5-7(a), the lighter shaded rectangle represents the inflated intersection due to the discretization and approximation for the efficient solution. The blue boxes represent the time steps when the supervisor intervenes. Although the supervisor intervenes, the last vehicle is not overridden, that is, the control input (black line in Figure 5-7(b)) remains equivalent to the desired input (blue dotted line in Figure 5-7(b)).

5-8 The same initial condition but a varying desired input are considered. The control input (black line in Figure 5-8(b)) maintains the similar level to the desired input (blue dotted line in Figure 5-8(b)).

5-9 Average cost on 60 samples of four vehicles. Random schedule (blue bars) represents the results from solving Problem 5.3 while Optimal schedule (red bars) represents the results from solving Problem 5.3 and Problem 5.4.
List of Algorithms

3.1 Verification of Problem 3.2 ........................................ 31
3.2 Solution of Problem 3.3 ........................................... 34
4.3 Solution of DEC(1|ri, pi = 1|Lmax,0) .......................... 40
4.4 Verification of Problem 4.1 ................................. 42
Chapter 1

Introduction

In the United States, the most critical and frequent vehicle collisions occur at or near intersections according to the National Motor Vehicle Crash Causation Survey (NMVCCS) [2]. In response to this, there have been a number of attempts within academia, industry, and government to reduce the number of accidents at intersections. One of the most conspicuous examples is Google's self-driving car, which has surpassed 700,000 autonomous driving miles without collisions, including at or near intersections [19]. However, the fully automated driving is still controversial [24]; one of the reasons is that people prefer having control of their vehicles. The alternative promising approach has been semi-autonomous driving, which guarantees the freedom of the drivers and takes over the controls only when safety constraints are violated [4,23].

In order to inform, warn, and eventually override human drivers to prevent crashes, the auto industry and the government have been developing vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication [1]. The potential for using multi-vehicle communication has stimulated research on networked vehicle systems to maintain safety since the 90's with the California PATH project [3], focusing mostly on highways. More recently, works employing job scheduling and optimal control [9,10,16,17] for intersection collision avoidance have appeared. Works that mostly focused on warning for collision avoidance at intersections have also appeared for general traffic scenarios [8,22]. Driver behavior estimation and classification near
intersections has also been of interest [5,13].

This paper exploits the V2I communication so that a supervisor embedded in a roadside infrastructure is able to measure the position and speed of all approaching vehicles and give commands to some of them, called controlled vehicles. The controlled vehicles are equipped with driver assist systems that enable the supervisor to communicate with the vehicles and override the human drivers. In this paper, the design of a centralized supervisor is described to prevent intersection collisions for a heterogeneous system, that is, in the presence of multiple controlled and uncontrolled vehicles. We thus extend the work of [10], which applies only to cases when all vehicles are controlled. To consider uncontrolled vehicles, a different problem formulation of safety verification and a different state prediction scheme are required. In particular, safety verification must consider all possible future states of uncontrolled vehicles to estimate when they will cross an intersection since the inputs for the uncontrolled vehicles are neither measured nor controlled.

An inserted idle-time (IIT) scheduling approach [15] is employed to detect upcoming collisions in the presence of uncontrolled vehicles. Depending on the outcome of safety verification, the supervisor intervenes if necessary. In general, computational complexity is a major concern in centralized control systems [6], and the IIT scheduling used in this paper is known to be NP-hard [14], that is, the IIT scheduling sometimes cannot be solved in short time. We address this problem by proposing an efficient scheduling approach that solves a relaxed problem. We quantify the extent of conservatism of the solution by determining an equivalent “Bad” set, which the trajectories of the system are forced to avoid. Based on this efficient solution, we further formulate optimization problems to generate a smoother input profile. To implement the optimization problems, we consider the corresponding discrete system dynamics and provide a more conservative Bad set.

This paper is organized as follows. In Chapter 2, we review the mathematical framework of scheduling problems and define the system. Chapter 3 presents two main problems, Verification and Supervisory, and provides the algorithms that exactly solve these problems. To address the computational complexity of these exact solutions,
approximate problems and efficient solutions are introduced in Chapter 4. To generate smoother control inputs, optimal approaches are presented in Chapter 5. The exact, efficient, and optimal approaches are followed by their simulation results. Conclusions and future works are also provided in Chapter 6.
Chapter 2

Basic Concepts

In this chapter, we introduce several basic concepts that are used through this paper. In section 2.1, mathematical background of scheduling and equivalent relation of problems are introduced. In section 2.2, we define an intersection with assumption that all cars are moving on pre-determined longitudinal paths that are converging to one point. In section 2.3, a dynamical system is modeled, and several notations and properties of system variables are given.

2.1 Mathematical Background

2.1.1 Scheduling

Scheduling problems are described by jobs, machines, and processing characteristics [11, p. 4]. Jobs represent tasks which have to be executed; machines represent scarce resources such as facilities where jobs are performed; and processing characteristics used in this paper are release times \( r_i \), deadlines \( d_i \), and process times \( p_i \). A schedule is a vector of starting times \( t_i \) for all jobs such that all constraints in a scheduling problem are satisfied. While most scheduling literature focuses on non-delay schedules, we consider a more general class of schedules which is called an inserted idle-time (IIT) schedule. This schedule is created by considering an IIT, which is an open time interval when the machine is deliberately held idle while at
At least one job is waiting to be processed [15]. If we say \((\bar{r}_\gamma, \bar{p}_\gamma)\) is an IIT, a necessary condition for an IIT schedule \(t_i\) to be feasible is for all \(i\),

\[(t_i, t_i + p_i) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset.\]

In this paper, we consider the scheduling problem DEC\((1|\text{r}|L_{\text{max}}|0)\), which aims to find a schedule that makes the maximum lateness \(L_{\text{max}} := \max t_i + p_i - d_i\) smaller than or equal to 0. This is stated in a formal manner as follows.

**Definition 2.1 (DEC\((1|\text{r}|L_{\text{max}}|0))\)**. Given a set of \(n\) jobs that have release times \(r_i\), deadlines \(d_i\), and process times \(p_i\) over one machine and given a set of \(m\) inserted idle-times \((\bar{r}_\gamma, \bar{p}_\gamma)\), determine whether there exists a schedule \(T = (t_1, \ldots, t_n)\) such that for all \(i \in \{1, \ldots, n\}\),

\[r_i \leq t_i \leq d_i - p_i,\]

for all \(i \neq j \in \{1, \ldots, n\}\)

\[t_i \leq t_j \Rightarrow t_i + p_i \leq t_j,\]

and for all \(\gamma \in \{1, \ldots, m\}\)

\[t_i \leq \bar{r}_\gamma \Rightarrow t_i + p_i \leq \bar{r}_\gamma,\]

\[t_i \geq \bar{r}_\gamma \Rightarrow t_i \geq \bar{p}_\gamma.\]

Notice that if such a schedule exists, \((t_i, t_i + p_i) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset\) for all \(i\) and \(\gamma\) because for all \(t^* \in (t_i, t_i + p_i)\), the last two constraints imply \(t^* < \bar{r}_\gamma\) or \(t^* > \bar{p}_\gamma\). The problem DEC\((1|\text{r}, p_i = 1|L_{\text{max}}|0)\) has the same constraints as in Definition 2.1 except with the addition of the constraint \(p_i = 1\). It has been shown that a problem with arbitrary process times such as DEC\((1|\text{r}|L_{\text{max}}|0)\) is NP-hard, while a problem with unit process times such as DEC\((1|\text{r}, p_i = 1|L_{\text{max}}|0)\) is solved in polynomial time [21]. The problem DEC\((1|\text{r}, p_i = 1|L_{\text{max}}|0)\) is used to address the complexity issue in Chapter 4.
Figure 2-1: An example of 3 vehicles approaching an intersection. The traffic intersection (left) is modeled as one point at which unidirectional paths intersect (right). Here, $\alpha_i$ and $\beta_i$ represent the location of the intersection on the longitudinal path $i$.

2.1.2 Equivalence

To solve a problem relying on the solution to a different problem, concepts of "reducibility" and "equivalence" [18] are introduced. In general, an instance of a problem is the information required to compute a solution to the problem, while satisfying all given constraints [12, p. 5]. A problem $P_1$ is reducible to a problem $P_2$ if for an instance $I_2$ of $P_2$ that can be constructed in polynomial-bounded time for any instance $I_1$ of $P_1$, $P_2$ accepts $I_2$ if and only if $P_1$ accepts $I_1$. We write $P_1 \preceq P_2$ if $P_1$ is reducible to $P_2$. If $P_2 \preceq P_1$ and $P_1 \preceq P_2$, $P_1$ and $P_2$ are equivalent.

2.2 Intersection Model

In this paper, we model the intersection as $n$ vehicles moving along unidirectional paths intersecting at one point (Figure 2-1). We assume that a subset of these vehicles can be controlled while the remaining vehicles cannot be controlled. We identify vehicles with a natural number from 1 to $n$. To distinguish between controlled and uncontrolled vehicles, we define a controlled set $C$ and an uncontrolled set $\bar{C}$, which
contain $n_c$ and $n_e$ elements, respectively, as follows:

$$ C := \{ i \in \{1, \ldots, n\} : \text{vehicle } i \text{ is controlled} \} $$

$$ \bar{C} := \{ \gamma \in \{1, \ldots, n\} : \text{vehicle } \gamma \text{ is uncontrolled} \}. $$

In this paper, $i$ usually denotes a controlled vehicle while $\gamma$ denotes an uncontrolled vehicle. For notational brevity, let $C = \{1, \ldots, n_c\}$ and $\bar{C} = \{n_c + 1, \ldots, n\}$ through this paper.

### 2.3 Dynamic Model

Let $x_i$ represent the dynamic state of vehicle $i$ with $y_i \in \mathbb{R}$ the position on its path. For $i \in C$, let $u_i$ represent the input to vehicle $i$ and for $\gamma \in \bar{C}$, let $d_\gamma$ represent a disturbance, that is, an unknown input to vehicle $\gamma$. Each controlled vehicle is modeled by

$$ \dot{x}_i = f_i(x_i, u_i), \quad y_i = h_i(x_i), \quad (2.1) $$

while each uncontrolled vehicle is modeled by

$$ \dot{x}_\gamma = f_\gamma(x_\gamma, d_\gamma), \quad y_\gamma = h_\gamma(x_\gamma). \quad (2.2) $$

where $x_i \in X_i \subseteq \mathbb{R}^r$, $u_i \in U_i \subseteq \mathbb{R}^s$, $y_i \in Y_i \subseteq \mathbb{R}$, $x_\gamma \in X_\gamma \subseteq \mathbb{R}^r$, $d_\gamma \in D_\gamma \subseteq \mathbb{R}^s$, and $y_\gamma \in Y_\gamma \subseteq \mathbb{R}$. The functional spaces of the piecewise continuous signals $u_i(t)$ and $d_\gamma(t)$ for $t \in [0, \infty)$ are $U_i$ and $D_\gamma$, respectively. The symbols $\mathbf{x}, \mathbf{u}, \mathbf{d}, \mathbf{y}$ denote aggregate vectors for these states, inputs, disturbances, and outputs. For instance $\mathbf{x} = (x_1, \ldots, x_n)$. The spaces of these vectors are denoted by $X, Y, U, D, U, D$. Notice that $X \subseteq (\mathbb{R}^r)^n$ and $Y \subseteq \mathbb{R}^n$ while $U \subset (\mathbb{R}^s)^{n_c}$ and $D \subset (\mathbb{R}^s)^{n_e}$. Moreover, the output function $h_i(x_i)$ and $h_\gamma(x_\gamma)$ are continuous and the derivative of the output is bounded, that is, $\dot{y}_i \in [\dot{y}_{i,m}, \dot{y}_{i,M}]$ with $\dot{y}_{i,m} > 0$. We assume that sets $U_i$ and $D_\gamma$ are intervals, that is, $U_i = [u_{i,m}, u_{i,M}]$ and $D_\gamma = [d_{\gamma,m}, d_{\gamma,M}]$. 
The parallel composition of (2.1) and (2.2) for all \( n \) vehicles describes the system dynamics:

\[
\dot{x} = f(x, u, d), \quad y = h(x).
\] (2.3)

We assume that this system has a unique solution that continuously depends on the input and the disturbance. The state and output of vehicle \( i \) at time \( t \) starting from \( x_i(t_0) \) with input \( u_i \) are denoted by \( x_i(t, u_i, x_i(t_0)) \) and \( y_i(t, u_i, x_i(t_0)) \), respectively. The corresponding aggregate state is denoted by \( x(t, u, d, x(t_0)) \) and the output by \( y(t, u, d, x(t_0)) \). For notational brevity, we omit an initial condition when \( t_0 = 0 \) and also omit the other arguments when they are not important. From now on, we consider \( t_0 = 0 \).

For any two states \( x_i, x'_i \in X_i \subseteq \mathbb{R}^r \), we say that \( x_i \leq x'_i \) if \( x_{ij} \leq x'_{ij} \) for all \( j \in \{1, \ldots, r\} \) where \( x_{ij} \) and \( x'_{ij} \) are the \( j \)-th elements of \( x_i \) and \( x'_i \), respectively. Similarly, for any two time signals \( u_i, u'_i \in \mathcal{U}_i \), we say that \( u_i \leq u'_i \) if \( u_i(t) \leq u'_i(t) \) for all \( t \geq 0 \). We assume that the trajectories \( y_i(t, u_i, x_i(0)) \) and \( y_\gamma(t, d_\gamma, x_\gamma(0)) \) are order preserving functions of their arguments. That is, if \( u_i \leq u'_i \) for \( u_i, u'_i \in \mathcal{U}_i \) and \( d_\gamma \leq d'_\gamma \) for \( d_\gamma, d'_\gamma \in \mathcal{D}_\gamma \), then \( y_i(t, u_i, x_i(0)) \leq y_i(t, u'_i, x_i(0)) \) and \( y_\gamma(t, d_\gamma, x_\gamma(0)) \leq y_\gamma(t, d'_\gamma, x_\gamma(0)) \) for all \( t \geq 0 \). Also, if \( x_i(0) \leq x'_i(0) \), then \( y_i(t, u_i, x_i(0)) \leq y_i(t, u_i, x'_i(0)) \) and if \( x_\gamma(0) \leq x'_\gamma(0) \), then \( y_\gamma(t, d_\gamma, x_\gamma(0)) \leq y_\gamma(t, d_\gamma, x'_\gamma(0)) \). In the same sense, the states \( x_i(t, u_i, x_i(0)) \) and \( x_\gamma(t, d_\gamma, x_\gamma(0)) \) are order preserving functions of their arguments.
Chapter 3

An Exact Approach

In this paper, two main problems are addressed: the Verification Problem and the Supervisory Problem. The Verification Problem determines whether the current drivers' inputs will lead to a collision. If this is the case, the Supervisory Problem overrides all controlled vehicles with safe inputs. In Section 3.1, these problems are formulated in a formal manner. An exact solution for these problems are given in Section 3.2. Finally, simulation results validate these solutions in Section 3.3 and show computational defects in these solutions.

3.1 Problem Statement

In this section, we formulate the two main problems, the Verification Problem and the Supervisory Problem. Also, the Inserted Idle Time Scheduling Problem is introduced, which is proved to be equivalent to the Verification Problem.

3.1.1 Verification Problem

As seen in Figure 2-1, an interval \((\alpha_i, \beta_i)\) is assigned to vehicle \(i\) for all \(i \in C \cup \bar{C}\). Let \(\alpha := \{\alpha_1, \ldots, \alpha_n\}\) and \(\beta := \{\beta_1, \ldots, \beta_n\}\). If two or more vehicles, at least one of which is a controlled vehicle, are inside the intersection at the same time, we consider that a collision occurs. Accordingly, we define the bad set \(B \subseteq Y\) as the set of output
configurations corresponding to a collision as follows:

\[ B := \{ y \in Y : y_i \in (\alpha_i, \beta_i) \text{ and } y_j \in (\alpha_j, \beta_j) \} \quad (3.1) \]

for some \( i \neq j \) such that \( i \in C \) and \( j \in C \cup \bar{C} \} \). \quad (3.2)

This implies that we focus on preventing collisions in which one or more controlled vehicles are involved. These include collisions between controlled and uncontrolled vehicles and collisions among controlled vehicles.

**Problem 3.1 (Verification Problem).** *Given an initial condition \( x(0) \), determine if there exists an input signal \( u \in U \) which guarantees that \( y(t, u, d) \notin B \) for all \( d \in D \) for all \( t \geq 0 \).*

Input \( u \in U \) ensures safety if and only if \( y(t, u, d) \notin B \) for all \( d \in D \) for all \( t \geq 0 \). An instance of Problem 3.1 is described by the initial condition \( x(0) \) and the parameters \( \Theta = \{ f, h, X, Y, U, D, \mathcal{D}, \alpha, \beta \} \). Thus, if there exists \( u \) which ensures safety for a given instance \( \{ x(0), \Theta \} \), then we say \( \{ x(0), \Theta \} \in \text{Problem 3.1} \).

To solve Problem 3.1, we adapt the IIT scheduling problem \( \text{DEC}(1|\tau_i|L_{\text{max}}, 0) \) in Definition 2.1 to our problem. In particular, the intersection plays the role of the machine, and the \( n_c \) vehicles are the jobs. Furthermore, we define release times \( R_i \), deadlines \( D_i \), process times \( P_i(T_i) \), and IITs \( (\bar{R}_i, \bar{P}_i) \) as follows.

**Definition 3.1.** *For all \( i \in C \) where \( y_i(0) < \alpha_i \), let*

\[ R_i := \min_{u_i \in U} \{ t \geq 0 : y_i(t, u_i) = \alpha_i \} \]

\[ D_i := \max_{u_i \in U} \{ t \geq 0 : y_i(t, u_i) = \alpha_i \} \]

*Given a non-negative real number \( T_i \), let*

\[ P_i(T_i) := \min_{u_i \in U} \{ t \geq 0 : y_i(t, u_i) = \beta_i \}
\]

*with constraint \( y_i(T_i, u_i) = \alpha_i \).*

26
For all $\gamma \in \mathcal{C}$ where $y_\gamma(0) < \alpha_\gamma$, let

$$\bar{R}_\gamma := \min_{d_t \in D_\gamma} \{t \geq 0 : y_\gamma(t, d_t) = \alpha_\gamma\},$$

$$\bar{P}_\gamma := \max_{d_t \in D_\gamma} \{t \geq 0 : y_\gamma(t, d_t) = \beta_\gamma\}.$$

For $i \in \mathcal{C}$, if $y_i(0) \geq \beta_i$, then set $R_i = 0, D_i = 0,$ and $P_i(T_i) = 0$. If $y_i(0) \geq \alpha_i$, then set $R_i = 0, D_i = 0,$ and $P_i(T_i) = \min_{u_t \in U_4} \{t : y_i(t, u_t) = \beta_i\}$. If the constraint $y_i(T_i, u_t) = \alpha_i$ is not satisfied, set $P_i(T_i) = \infty$. For $\gamma \in \mathcal{C}$, if $y_\gamma(0) \geq \beta_\gamma$, set $\bar{R}_\gamma = 0$ and $\bar{P}_\gamma = 0$. If $y_\gamma(0) \geq \alpha_\gamma$, set $\bar{R}_\gamma = 0$ and $\bar{P}_\gamma = \max_{d_t \in D_\gamma} \{t : y_\gamma(t, d_t) = \beta_\gamma\}$.

These variables are well-defined since it is assumed that the system (2.3) has a unique solution and $\dot{y}_i > 0$. Notice that $R_i, D_i, \bar{R}_\gamma, \bar{P}_\gamma$ for all $i \in \mathcal{C}$ and $\gamma \in \mathcal{C}$ are fixed once an initial condition is provided. We can view $(\bar{R}_\gamma, \bar{P}_\gamma)$ for $\gamma \in \mathcal{C}$ as an IIT because the intersection needs to be kept empty as long as an uncontrolled vehicle can occupy it. We introduce the following scheduling problem.

**Problem 3.2 (IIT Scheduling Problem).** Given an initial condition $x(0)$, determine whether there exists a schedule $T = (T_1, \ldots, T_n)$ where $\mathcal{C} = \{1, \ldots, n_c\}$, such that for all $i \in \mathcal{C},$

$$R_i \leq T_i \leq D_i,$$  \hspace{1cm} (3.3)

for all $i \neq j \in \mathcal{C},$

$$T_i \leq T_j \Rightarrow P_i(T_i) \leq T_j,$$  \hspace{1cm} (3.4)

for all $i \in \mathcal{C}$ and $\gamma \in \mathcal{C},$

$$T_i \leq \bar{R}_\gamma \Rightarrow P_i(T_i) \leq \bar{R}_\gamma,$$  \hspace{1cm} (3.5)

$$T_i \geq \bar{R}_\gamma \Rightarrow T_i \geq \bar{P}_\gamma.$$  \hspace{1cm} (3.6)

This problem is also described by an instance $I = \{x(0), \Theta\}$. We say $\{x(0), \Theta\} \in$ Problem 3.2 if there exists a schedule $T$ that satisfies all constraints in Problem 3.2.

A feasible schedule for Problem 3.2 ensures $(T_i, P_i(T_i)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = \emptyset$ by (3.5) and (3.6). The IIT scheduling determines if a sequence of controlled vehicles exists such
that controlled vehicles cross an intersection before or after IIT. We now prove that
Problem 3.2 is equivalent to Problem 3.1 and then propose a solution to Problem 3.2
in Section 3.2.

**Theorem 3.1.** Problem 3.1 and Problem 3.2 are equivalent.

**Proof.** We prove that the Verification Problem (Problem 3.1) is reducible to the IIT
Scheduling Problem (Problem 3.2) and vice versa. Notice that Problem 3.1 and 3.2
have the same instance \( I \in \{x(0), \Theta \} \). Thus, equivalence between the two problems
follows by showing that for any given instance \( I = \{x(0), \Theta \} \),

\[
I \in \text{Problem 3.1} \Leftrightarrow I \in \text{Problem 3.2}.
\]

(\( \Rightarrow \)) Given an initial condition \( x(0) \), there is an input \( \tilde{u} \) which ensures safety, that is,\( y(t, \tilde{u}, d) \notin B \) for all \( d \) and for all \( t \geq 0 \). For \( y_i(0) < \alpha_i \) for \( i \in C \), define \( T_i \) as the time \( t \) when \( y_i(t, \tilde{u}_i) = \alpha_i \), and for \( y_i(0) < \beta_i \), define \( \tilde{T}_i(T_i) \) as the time when \( y_i(t, \tilde{u}_i) = \beta_i \).
If \( y_i(0) \geq \alpha_i \), set \( T_i = 0 \). Set \( \tilde{T}_i(T_i) = 0 \) if \( y_i(0) \geq \beta_i \).

By definitions of \( R_i \) and \( D_i \), \( R_i \leq T_i \leq D_i \) (Condition (3.3)). Suppose \( T_i \leq T_j \) for
some \( i \neq j \in C \). Since \( \tilde{u}_i \) and \( \tilde{u}_j \) prevent an intersection collision between vehicles \( i \)
and \( j \), when \( y_j(t, \tilde{u}_j) = \alpha_j \), we have that \( y_i(t, \tilde{u}_i) \geq \beta_i \). Since \( y_i > 0 \) by assumption,
\( \tilde{T}_i(T_i) \leq T_j \). By definition of \( P_i(T_i) \), it follows that \( P_i(T_i) \leq \tilde{T}_i(T_i) \leq T_j \) (Condition
(3.4)). Also, with the assumption that the input \( \tilde{u} \) ensures safety for all \( d \), if \( T_i \leq \tilde{R}_\gamma \)
for \( i \in C, \gamma \in \bar{C} \), when \( \max_{d_\gamma} y_\gamma(t, d_\gamma) = \alpha_\gamma \), it must be true that \( y_i(t, \tilde{u}_i) \geq \beta_i \). Since \( y_\gamma(t) \) is increasing with time, \( \{t : \max_{d_\gamma} y_\gamma(t, d_\gamma) = \alpha_\gamma \} \) becomes \( \min_{d_\gamma} \{t : y_\gamma(t, d_\gamma) = \alpha_\gamma \} \) which is equal to \( \tilde{R}_\gamma \) by definition. Combining this with the definition of \( P_i(T_i) \),
we have that \( P_i(T_i) \leq \tilde{T}_i(T_i) \leq \tilde{R}_\gamma \) (Condition (3.5)). If \( T_i \geq \tilde{R}_\gamma \), when \( y_i(t, \tilde{u}_i) = \alpha_i \),
the vehicle \( \gamma \) must stay away from the intersection, i.e., \( \min_{d_\gamma} y_\gamma(t, d_\gamma) \geq \beta_\gamma \). Again,
because of the increasing property of the output with respect to \( t \), \( T_i \geq \tilde{P}_\gamma \) (Condition
(3.6)).

\( (\Leftarrow) \) Given an initial condition \( x(0) \), there exists a schedule \( T \in \mathbb{R}^n \) that satisfies
all conditions of Problem 3.2. We start assuming that \( y_i(0) < \alpha_i \) for all \( i \in C \cup \bar{C} \),
which does not break generality since vehicles after the intersection are not of interest.
According to Lemma 5.1 in [10], if \( T_i \in [R_i, D_i] \) for \( y_i < \alpha_i \), there exists an input \( u_i \in U_i \) such that \( y_i(T_i, u_i) = \alpha_i \). Then, we can construct \( u_i \in U_i \) such that \( y_i(T_i, u_i) = \alpha_i \) for all \( i \in C \).

Now, we show that these inputs \( u_i \) for \( i \in C \) ensure safety, if \( T_i \) satisfy conditions (3.3)-(3.6). When \( T_i \leq T_j \) for \( i \neq j \in C \), condition (3.4) indicates \( P_i(T_i) \leq T_j \), which implies that when \( y_i(t, u_i) = \beta_i \), we have \( y_j(t, u_j) \leq \alpha_j \) so that \( u_i \) and \( u_j \) prevent collisions between controlled vehicles. From condition (3.5), when \( T_i \leq \bar{R}_\gamma \) for \( i \in C, \gamma \in \bar{C}, P_i(T_i) \leq \bar{R}_\gamma \). This implies that when \( \max_{d_\gamma} y_\gamma(t, d_\gamma) = \alpha_\gamma \), we have \( y_i(t, u_i) \geq \beta_i \), that is, \( y_\gamma(t, d_\gamma) \leq \alpha_\gamma \) and \( y_i(t, u_i) \geq \beta_i \). If \( T_i \geq \bar{R}_\gamma \), condition (3.6) indicates that when \( y_i(t, u_i) = \alpha_i \), we have \( \min_{d_\gamma} y_\gamma(t, d_\gamma) \geq \beta_\gamma \). Thus, when \( y_i(t, u_i) \leq \alpha_i \), it is true for all \( d_\gamma \) that \( y_\gamma(t, d_\gamma) \geq \beta_\gamma \). Hence, \( u_i \) for \( i \in C \) prevent collisions between controlled and uncontrolled vehicles for all \( d_\gamma \).

3.1.2 Supervisory Problem

We now design a supervisor operating in discrete time. At each time step \( k\tau \), a current state \( x(k\tau) \in X \) and a desired input \( a_k \in U \) for the controlled vehicles are given, where the desired input represents the input applied by a driver. Then, the supervisor returns an input based on the partial state prediction for the next step. We define the \( j \)-th entry of this partial state prediction as follows:

\[
[x_{k+1}(a_k)]_j := \begin{cases} 
  x_j(\tau, a_{k,j}, x_j(k\tau)) & j \in C \\
  x_j(k\tau) & j \in \bar{C}
\end{cases}
\]

where \( a_{k,j} \in U_j \) for \( j \in C \) is the \( j \)-th entry of the vector \( a_k \), and it is known since it is for a controlled vehicle. Notice that the current disturbance \( d \in D \) does not contribute to the partial state prediction since uncontrolled vehicles are not equipped with driver assist systems that can monitor the input of the drivers. This is why we called \( x_{k+1}(a_k) \) the partial state prediction.

Because a supervisor interacts with human operators, it has to be least restrictive in the sense that it intervenes only if a collision is guaranteed to occur in the future.
for some $d \in D$. To describe future trajectories, define a continuous input profile $u_k^\infty(t)$ defined for $t \in [k\tau, \infty)$ and let $u_k(t)$ be $u_k^\infty(t)$ restricted to $t \in [k\tau, (k+1)\tau]$. During this time period, let $u_k(t) = a_k$, where $a_k$ is the desired input at $t = k\tau$ and is considered constant on $[k\tau, (k+1)\tau]$. Similarly, let $u_{k,\text{safe}}^\infty(t)$ be a safe input defined for $t \in [k\tau, \infty)$ and let $u_{k,\text{safe}}(t)$ be $u_{k,\text{safe}}^\infty(t)$ restricted to $t \in [k\tau, (k+1)\tau]$. We now formalize the Supervisory Problem.

**Problem 3.3 (Supervisory Problem).** Design a supervisor $s(x(k\tau), a_k)$ at time step $k\tau$ such that

$$s(x(k\tau), a_k) = \begin{cases} u_k & \text{if } \exists u_k^\infty : y(t, u_k^\infty, d) \notin B \\ & \text{for all } d \in D \text{ for } t \geq k\tau \\ u_{k,\text{safe}} & \text{otherwise,} \end{cases}$$

and such that it is non-blocking: if $s(x(k\tau), a_k) \neq \emptyset$, then for any $a_{k+1}$, $s(x((k+1)\tau), a_{k+1}) \neq \emptyset$.

### 3.2 Problem Solutions

By virtue of Theorem 3.1, we can solve the Verification Problem by solving the IIT Scheduling Problem. In this section, we thus focus on providing an exact solution to the IIT Scheduling Problem, which we then use to solve the Supervisory Problem.

#### 3.2.1 Solution of the Verification Problem

Assume, without loss of generality, that $y_i \geq \alpha_i$ for $m$ controlled vehicles. Let $\mathcal{P}$ be the set of all permutations of controlled vehicles such that $y_i < \alpha_i$. Then, each element of $\mathcal{P}$ is an $(n_c - m)$-tuple denoted by $\pi$. Let $\bar{\pi}$ be an $n_c$-tuple composed of all $\gamma \in \bar{C}$ in an increasing order of $\bar{R}_{\gamma}$. The notation $\pi_i$ represents the $i$-th element of $\pi$, and $\bar{\pi}_i$ represents the $i$-th element of $\bar{\pi}$. Parameter $\delta$ represents a time delay, as explained in Section 3.2.2.

Algorithm 3.1 with $\delta = 0$ solves Problem 3.2 (IIT Scheduling Problem), and, hence, Problem 3.1 (Verification Problem) by Theorem 3.1. Also, notice that the
Algorithm 3.1 Verification of Problem 3.2

procedure EXACTSOLUTION(x(0), δ)
    if y(0) ∈ B then return {0, no}
    for all i ∈ C and γ ∈ C do
        given x₁(0) calculate Rᵢ, Dᵢ
        given xᵣ(0) calculate Rᵣ, Pᵣ
        $Rᵣ \leftarrow \max(Rᵣ - δ, 0)$
        $Pᵣ \leftarrow \max(Pᵣ - δ, 0)$
    for i ∈ C such that yᵢ(0) ≥ αᵢ do
        $Tᵢ \leftarrow 0$
        calculate $Pᵢ(Tᵢ)$ and $P_{max} \leftarrow \max Pᵢ(Tᵢ)$
    for all $π ∈ P$ do
        for j ← 1 to $n_c - m$ do
            $T_{πj} \leftarrow \max(R_{πj}, P_{max})$ for $j = 1$
            $T_{πj} \leftarrow \max(R_{πj}, P_{πj-1}(T_{πj-1}))$
        for $r ← 1$ to $n_π$ do
            if $T_{πj} ≥ R_{πr}$ then
                $T_{πj} \leftarrow \max(T_{πj}, P_{πr})$
                given $T_{πj}$ calculate $P_{πj}$
            else if $P_{πj} ≥ R_{πr}$ then
                $T_{πj} \leftarrow P_{πr}$
                given $T_{πj}$ calculate $P_{πj}$
        if $Tᵢ ≤ Dᵢ$ for all $i ∈ C$ then
            return {T, yes}
running time of Algorithm 3.1 increases in a factorial manner with \((n_c - m)\) since the algorithm checks all permutations in \(\mathcal{P}\) to return no. This is consistent with the fact that Problem 3.2 is known to be NP hard [14].

**Example 3.1.** Suppose we have \(C = \{1, 3, 4\}\) and \(\bar{C} = \{2, 5\}\) that are modeled for simplicity as \(f(x_i, u_i) = u_i, h(x_i) = x_i\) for \(i \in C\), and \(f(x_\gamma, d_\gamma) = d_\gamma, h(x_\gamma) = x_\gamma\) for \(\gamma \in \bar{C}\). A given initial condition is \(x(0) = (44, 26, 20, 5, 2)\) with \(\delta = 0\) with parameters \(u_m = 3, u_M = 15, d_m = 6, d_M = 12\) and \((\alpha_i, \beta_i) = (50, 53)\). The release times become \((R_1, R_3, R_4) = (\frac{2}{5}, 2, 3)\), the deadlines \((D_1, D_3, D_4) = (2, 10, 15)\), and the idle-times \((\bar{R}_2, \bar{R}_5) = (2, 4)\), and \((\bar{P}_2, \bar{P}_5) = (4\frac{1}{2}, 8\frac{1}{2})\).

Consider \(\pi = (1, 3, 4)\) and \(\bar{\pi} = (2, 5)\). For \(j = 1\) and \(\pi_1 = 1\), we have \(T_1 = R_1 = \frac{2}{5}\), and \(P_1 = T_1 + \frac{\beta_{\bar{\pi}} - \alpha_1}{u_M} = \frac{3}{5}\). Since \(T_1 < \bar{R}_2 < \bar{R}_5\) and \(P_1 < \bar{R}_2 < \bar{R}_5\), we determine that \(T_1 = \frac{2}{5}\) and \(P_1 = \frac{3}{5}\). For \(j = 2\) and then \(\pi_2 = 3\), we have \(T_3 = \max(R_3, P_1) = 2\) and \(P_3 = 2\frac{1}{5}\). However, for \(r = 1\), we have \(\bar{\pi}_1 = 2\) and \(T_3 \geq \bar{R}_2\). Thus, \(T_3 = \max(T_3, \bar{P}_2) = 4\frac{1}{2}\). This leads to \(P_3 = 4\frac{7}{10}\). For \(r = 2\), we have \(\bar{\pi}_2 = 5\). Because \(T_3 < \bar{R}_5\) and \(P_3 > \bar{R}_5\), we have \(T_3 = \bar{P}_5 = 8\frac{1}{2}\) and \(P_3 = 8\frac{7}{10}\). Finally, for \(j = 3\), we have \(\pi_3 = 4\) so that \(T_4 = \max(R_4, P_3) = 8\frac{7}{10}\) and \(P_4 = 8\frac{9}{10}\). Even though for \(r = 1\) we have \(T_4 \geq \bar{R}_1\), notice that \(T_4\) does not change because \(\max(T_4, \bar{R}_1) = T_4\). This is the same for \(r = 2\).

Since \(T_i \leq D_i\) for all \(i \in \{1, 3, 4\}\), we have found a set of schedule \(\Upsilon = (\frac{2}{5}, 8\frac{1}{2}, 8\frac{7}{10})\). If there was no feasible schedule for this \(\pi\), Algorithm 3.1 would try the other permutations such as \(\pi = (1, 4, 3)\).

### 3.2.2 Solution of the Supervisory Problem

A supervisor is designed to override controlled vehicles with a safe input \(u_{k, safe}\) if a desired input \(a_k\) leads to an intersection collision at some future time. To achieve this goal, we define an input operator for \(y_i(0) < \alpha_i\) for \(i \in C\) as follows:

\[
\sigma_i(x_i(0), T_i) := \arg \inf_{u_i \in U_i} \{t \geq 0 : y_i(t, u_i) = \beta_i \text{ for } i \in C \text{ with constraint } y_i(T_i, u_i) = \alpha_i\}.
\] (3.8)
This operator returns an input \( u_i \in U_i \) for \( i \in C \) such that \( y_i(t, u_i) \) reaches \( \alpha_i \) at \( T_i \) and \( \beta_i \) at \( P_i(T_i) \). Set \( \sigma_i(x_i(0), T_i) = u_{i,M} \) if \( y_i(0) \geq \alpha_i \), and set \( \sigma_i(x_i(0), T_i) = 0 \) if \( y_i(0) \geq \beta_i \). If the constraint \( y_i(T_i, u_i) = \alpha_i \) cannot be satisfied, set \( \sigma_i(x_i(0), T_i) = \emptyset \). Let \( \sigma(x(0), T) \) be the parallel composition of (3.8) for all \( i \in C \). Then, \( \sigma(x(0), T) \) indicates the safe input \( u_{k,\text{safe}}^\infty \) introduced in Section 3.1.

At each time step, Algorithm 3.1 determines whether a predicted state from a desired input leads to an intersection collision. To account for the fact that in the partial state prediction \( x_{k+1}(a_k) \) we do not predict the states for the uncontrolled vehicles since their inputs are not known, a time delay \( \tau \) is considered for vehicle \( \gamma \) with \( \gamma \in \tilde{C} \). In particular, let \((\hat{R}_{\gamma}, \hat{P}_{\gamma})\) denote an IIT for a given initial condition \( x_{\gamma}(k\tau) \). That is,

\[
\hat{R}_{\gamma} := \min_{d_{\gamma} \in D_{\gamma}} \{ t : y_{\gamma}(t, d_{\gamma}, x_{\gamma}(k\tau)) = \alpha_{\gamma} \}
\]

and

\[
\hat{P}_{\gamma} := \max_{d_{\gamma} \in D_{\gamma}} \{ t : y_{\gamma}(t, d_{\gamma}, x_{\gamma}(k\tau)) = \beta_{\gamma} \}.
\]

Also, let \((\hat{R}_{\gamma}^{k+1}, \hat{P}_{\gamma}^{k+1})\) denote the predicted idle-time defined as follows:

\[
\hat{R}_{\gamma}^{k+1} := \min_{d_{\gamma}} \{ t : y_{\gamma}(t, d_{\gamma}, x_{\gamma}(k\tau)) = \alpha_{\gamma} \}
\]

and

\[
\hat{P}_{\gamma}^{k+1} := \max_{d_{\gamma}} \{ t : y_{\gamma}(t, d_{\gamma}, x_{\gamma}(k\tau)) = \beta_{\gamma} \}.
\]

From these definitions, it can be verified that

\[
(\hat{R}_{\gamma}^{k+1}, \hat{P}_{\gamma}^{k+1}) = (\hat{R}_{\gamma}^{k} - \tau, \hat{P}_{\gamma}^{k} - \tau). \tag{3.9}
\]

We propose Algorithm 3.2 to solve the Supervisory Problem (Problem 3.3). Notice that Algorithm 3.1 takes \( \delta = 0 \) when \( x(0) \) is a measured state and \( \delta = \tau \) when \( x(0) \) is a partial state prediction. The next lemmas and theorem prove that Algorithm 3.2 solves Problem 3.3.

**Lemma 3.1.** If \textsc{ExactSolution}(\( x(k\tau), 0 \)) = \{T, yes\}, then \( u := \sigma(x(k\tau), T) \neq \emptyset \).
Algorithm 3.2 Solution of Problem 3.3

procedure $s(x(k\tau), a_k)$

$$u_k(t) \leftarrow a_k \quad \forall t \in [k\tau, (k+1)\tau]$$

$\{T, answer\} \leftarrow$ EXACTSOLUTION($x_{k+1}(u_k), \tau$)

if answer = yes and $y(t, u_k, d) \not\in B$ for all $d \in D$ for all $t \in [k\tau, (k+1)\tau]$ then

$u_{k+1, safe}^\infty \leftarrow \sigma(x_{k+1}(u_k), T)$

$u_{k+1, safe} \leftarrow u_{k+1, safe}^\infty$ restricted to $[k\tau, (k+1)\tau]$.

return $u_k$

else

$\{T, answer\} \leftarrow$ EXACTSOLUTION($x_{k+1}(u_{k, safe}), \tau$)

$u_{k+1, safe}^\infty \leftarrow \sigma(x_{k+1}(u_{k, safe}), T)$

$u_{k+1, safe} \leftarrow u_{k+1, safe}^\infty$ restricted to $[k\tau, (k+1)\tau]$.

return $u_{k, safe}$

Moreover, EXACTSOLUTION($x_{k+1}(u_k), \tau$) = $\{T, yes\}$ where $u_k$ restricts $u$ to $[k\tau, (k+1)\tau]$.

Proof. Given an initial condition $x(k\tau)$, if a schedule $T$ exists, then $\{x(k\tau), \Theta\} \in$ Problem 3.1 by Theorem 3.1, indicating that there exists an input $u \in U$ such that $y(t, u, d, x(k\tau)) \not\in B$ for all $d$ for $t \in [k\tau, \infty)$. Thus, $\sigma(x(k\tau), T) \neq \emptyset$. Let $\bar{u}$ be $u$ restricted to $[(k+1)\tau, \infty)$. By assumption, the schedule $T$ is feasible, which implies that it satisfies the conditions that $T_i \in [R_i, D_i]$ for all $i \in C$, that $(T_i, P_i(T_i))$ does not intersect each other, and that $(T_i, P_i(T_i)) \cap (\bar{R}_{i, \gamma}^k, \bar{P}_{i, \gamma}^k) = \emptyset$ for all $\gamma \in \bar{C}$. Define $T'$ as $T'_i = \{t : y_i(t, \bar{u}_i, [x_{k+1}(u_k)]_i) = a_i\}$ for all $i \in C$ where $\bar{u}_i$ is the entry of $\bar{u}$ corresponding to vehicle $i$. Then $T'_i = T_i - \tau$ and $P'_i(T_i) = P_i(T_i) - \tau$ since $u_k$ equals $u$ on $[k\tau, (k+1)\tau]$. Consequently, intervals $(T'_i, P'_i(T'_i))$ do not overlap for all $i \in C$ and $(T'_i, P'_i(T'_i)) \cap (\bar{R}_{i+1}^k, \bar{P}_{i+1}^k) = \emptyset$ for all $\gamma \in \bar{C}$ due to (3.9). Also $T'_i \in [R'_i, D'_i]$ by construction. Therefore, EXACTSOLUTION($x_{k+1}(u_k), \tau$) has a feasible schedule $T'$ and thus returns yes.

Lemma 3.2. Given a desired input $a_k$, if EXACTSOLUTION($x_{k+1}(a_k), \tau$) = $\{T, yes\}$, then $\sigma(x_{k+1}(a_k), T) \neq \emptyset$.

Proof. Define $\hat{x}_{k+1}(a_k)$ such that its $j$-th element $[\hat{x}_{k+1}(a_k)]_j$ is either $x_j(\tau, a_{k,j}, x_j(k\tau))$ for $j \in C$ where $a_{k,j}$ is the $j$-th element of $a_k$, or $x_j(\tau, d_j, x_j(k\tau))$ for $j \in \bar{C}$ for any
disturbance \( d_j \in [d_{j,m}, d_{j,M}] \) on \([k\tau, (k + 1)\tau]\). Let \((\hat{R}^{k+1}_\gamma, \hat{P}^{k+1}_\gamma)\) denote the IIT corresponding to \([\hat{x}_{k+1}(a_k)]_\gamma\). Because the state is an order preserving function of the disturbance, we have

\[
x_\gamma(\tau, d_{\gamma,m}, x_\gamma(k\tau)) \leq [\hat{x}_{k+1}(a_k)]_\gamma \leq x_\gamma(\tau, d_{\gamma,M}, x_\gamma(k\tau)).
\]

Also, since the output is an order preserving function with respect to the initial condition, we have

\[
y_\gamma(t, d_{\gamma}, [\hat{x}_{k+1}(a_k)]_\gamma) \leq y_\gamma(t, d_{\gamma}, x_\gamma(\tau, d_{\gamma,M}, x_\gamma(k\tau))),
\]

and

\[
y_\gamma(t, d_{\gamma}, x_\gamma(\tau, d_{\gamma,m}, x_\gamma(k\tau))) \leq y_\gamma(t, d_{\gamma}, [\hat{x}_{k+1}(a_k)]_\gamma).
\]

These two inequalities and the definitions of \(R^{k+1}_\gamma\) and \(P^{k+1}_\gamma\) further imply that since \(\dot{y}_\gamma > 0\) by assumption, \(\hat{R}^{k+1}_\gamma \geq R^{k+1}_\gamma\) and \(\hat{P}^{k+1}_\gamma \geq P^{k+1}_\gamma\), respectively. Thus, we have that \((\hat{R}^{k+1}_\gamma, \hat{P}^{k+1}_\gamma) \subseteq (R^{k+1}_\gamma, P^{k+1}_\gamma)\).

If \(\text{EXACTSOLUTION}(x_{k+1}(a_k), \tau)\) returns \text{yes}, a schedule \(T\) satisfies that \(T_i \in [R_i, D_i]\), that \((T_i, P_i(T_i))\) does not intersect each other for all \(i \in C\), and that for all \(\gamma \in \tilde{C}\), \((T_i, P_i(T_i)) \cap (\hat{R}^{k+1}_\gamma, \hat{P}^{k+1}_\gamma) = \emptyset\). The last condition implies that \((T_i, P_i(T_i)) \cap (\hat{R}^{k+1}_\gamma, \hat{P}^{k+1}_\gamma) = \emptyset\). Therefore, this schedule \(T\) makes \(\text{EXACTSOLUTION}(x_{k+1}(a_k), 0)\) return \text{yes}. From Lemma 3.1, this implies \(\sigma(\hat{x}_{k+1}(a_k), T) \neq \emptyset\). Notice that \([\hat{x}_{k+1}(a_k)]_i\) is equal to \([x_{k+1}(a_k)]_i\) for all \(i \in C\). Thus, \(\sigma_i([x_{k+1}(a_k)]_i, T_i) = \sigma_i([\hat{x}_{k+1}(a_k)]_i, T_i)\).

Since \(\sigma(x_{k+1}(a_k), T)\) is composed of \(\sigma_i([x_{k+1}(a_k)]_i, T_i)\) for all \(i \in C\), it follows that \(\sigma(x_{k+1}(a_k), T) \neq \emptyset\).

**Theorem 3.2.** The supervisor \(s(x(k\tau), a_k)\) defined in Algorithm 3.2 solves Problem 3.3.

**Proof.** If \(\text{EXACTSOLUTION}(x_{k+1}(a_k), \tau)\) returns \text{yes}, by Lemma 3.2 there exists an input signal \(\sigma(x_{k+1}(a_k), T) \neq \emptyset\) that ensures safety. Thus, Algorithm 3.2 returns \(u_k\) which is identical to \(a_k\). If \(\text{EXACTSOLUTION}(x_{k+1}(a_k), \tau)\) returns \text{no}, Algorithm 3.2
returns \( u_{k,\text{safe}} \).

We next show that Algorithm 3.2 is non-blocking. We use mathematical induction to prove this. Suppose at \( t = k\tau \), \( s(x(k\tau), a_k) = u_{k,\text{out}} \) and \( u_{k,\text{out}} \neq \emptyset \). Then we must prove that \( s(x((k+1)\tau), a_{k+1}) \neq \emptyset \) for any \( a_{k+1} \). Notice from Algorithm 3.2 that this is possible when \( u_{k+1,\text{safe}} \) exists, that is, \( u_{k+1,\text{safe}} = \sigma(x_{k+1}(u_{k,\text{out}}), T) \neq \emptyset \).

As seen in Algorithm 3.2, \( u_{k,\text{out}} \) is either \( u_k \) or \( u_{k,\text{safe}} \). In the former case, \( \text{EXACTSOLUTION}(x_{k+1}(u_k), \tau) = \{T, \text{yes}\} \), and by Lemma 3.2, \( \sigma(x_{k+1}(u_k), T) \) is non-empty. In the latter case, because we assume that \( u_{k,\text{safe}} \) exists, \( \text{EXACTSOLUTION}(x_{k+1}(u_{k,\text{safe}}), \tau) \) returns yes by Lemma 3.1. In turn, Lemma 3.2 implies that \( u_{k+1,\text{safe}} \) exists. Therefore, the supervisor is non-blocking. \( \square \)

### 3.3 Simulation Results

Consider the design of the supervisor \( s \), which is described in Section 3.2, for multiple vehicles converging to an intersection in the presence of uncontrolled vehicles. The dynamic state of each vehicle is \( x_i = (p_i, v_i) \) where \( p_i \) is the position and \( v_i \) is the speed. With an input \( u_i \) and a disturbance \( d_\gamma \) for \( i \in C \) and \( \gamma \in C \), the longitudinal dynamics of each vehicle are modeled as follows:

\[
\begin{align*}
\dot{p}_i &= v_i, \\
\dot{v}_i &= \begin{cases} 
  u_i & v_i \in (v_{i,m}, v_{i,M}) \\
  0 & \text{otherwise,}
\end{cases} \\
\dot{p}_\gamma &= v_\gamma, \\
\dot{v}_\gamma &= \begin{cases} 
  d_\gamma & v_i \in (v_{\gamma,m}, v_{\gamma,M}) \\
  0 & \text{otherwise.}
\end{cases}
\end{align*}
\]

We implement the algorithms using MATLAB with a time step \( \tau = 0.1\)s. Each vehicle has the same parameters: \((\alpha_i, \beta_i) = (50m, 53m)\) and \([v_{i,m}, v_{i,M}] = [3m/s, 15m/s]\). Also, the input \( u_i \in [-2m/s^2, 2m/s^2] \) for \( i \in C \) and the disturbance \( d_\gamma \in [-1m/s^2, 1m/s^2] \) for \( \gamma \in C \). We consider the desired input as \( 1m/s^2 \) for all controlled vehicles and construct the disturbance using a random number generator in MATLAB.

In Figure 3-1 and Figure 3-2, Algorithm 3.2 is simulated, based on EXACTSOLU-
Figure 3-1: Positions of 5 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by the supervisor in Section 3.2 (Algorithm 3.2). The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The shaded region represents the location of the intersection.

Notice that the supervisor guarantees that only one vehicle crosses the shaded region at a time, that is, the trajectories avoid the bad set. The overall simulation time when using the exact supervisor in Figure 3-2 is 8000 seconds (7.3 seconds per iteration on average). This computation time is greater than the time step \( \tau = 0.1s \) so that this algorithm is unable to be implemented in a real time.
Figure 3-2: Positions of 8 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by the supervisor $s$ in Section 3.2 (Algorithm 3.2).
Chapter 4

An Efficient Approach

As seen from Chapter 3, Problem 3.1 is NP-hard. To address this computational complexity problem, we introduce a relaxed version of Problem 3.2, called the Relaxed Scheduling Problem, which can be solved in polynomial time. Based on this scheduling problem, we construct a modified version of Problem 3.1, called the Relaxed Verification Problem in Section 4.1. The corresponding supervisor, called the Efficient Supervisor, is designed in Section 4.2. Finally, in Section 4.3, simulation results verify that computation complexity problem is resolved by this efficient approach.

4.1 Relaxed Verification Problem

To formulate the Relaxed Verification Problem, we first introduce the Relaxed Scheduling Problem, whose solution is solved in polynomial time as can be found from the scheduling literature. Then, we modify Problem 3.1 by inflating the bad set in light of the relaxation for the Relaxed Scheduling Problem.

4.1.1 Relaxed Scheduling Problem

Garey et al. [14] proposed an efficient algorithm to solve DEC(1|r_i, p_i = 1|L_{max},0) in \(O(n^2)\) where \(n\) is the number of jobs. To design the algorithm, they introduce the
concept of a forbidden region \( F \), a time interval when no task is allowed to start to have a feasible schedule. Let \( F_k \subseteq \mathbb{R} \) be the \( k \)-th interval in \( F \). Algorithm 4.3 is adapted from Algorithm A in [14]. The difference between these two algorithms is that Algorithm A in [14] initially declares \( F \) an empty set, while here we allow to introduce \( F \) arbitrarily.

Example 4.1. Consider 3 jobs on a machine with \( r = (7,7,8) \), \( d = (12,12,10) \), and unit process time. An initially declared forbidden region is \( F = \{(8\frac{1}{2},10\frac{1}{2})\} \). The increasing order of \( r \) is \((1,2,3)\), so Algorithm 4.3 starts from \( i = 3 \). For \( d_1, d_2, d_3 \geq d_3 \), since \( d_1 - 1 = d_2 - 1 \notin F \), we have \( c_1 = 11, c_2 = 11 \), while \( c_3 = \inf(F_1) = 8\frac{1}{2} \) because \( d_3 - 1 \notin F_1 \). Since \( r_2 < r_3 \), we find \( c = 8\frac{1}{2} \) which is inside \([r_3, r_3 + 1)\), and thus we update \( F = \{(7\frac{1}{2},8),(8\frac{1}{2},10\frac{1}{2})\} \). For \( i = 2 \), we have \( d_1, d_2 \geq d_2 \) and \( c_1 = c_2 = 8\frac{1}{2} \), and neither \( i \neq 1 \) nor \( r_1 < r_2 \). For \( i = 1 \), since \( c_1 = c_2 = 7\frac{1}{2} \), we obtain \( F = \{(6\frac{1}{2},7),(7\frac{1}{2},8),(8\frac{1}{2},10\frac{1}{2})\} \). Then, the next step allocates a schedule to each job.
For it = 1, we have s = 7 and t₁ = 7. For it = 2, we have that j = 3 since d₃ is the least due date and r₃ ≤ s, we have t₃ = 8. For it = 3, s ∈ F₃ so that s becomes 10½; thus t₂ = 10½.

Algorithm 4.3 has a complexity that is polynomial with respect to the number of jobs n. We next relax Problem 3.2 to DEC(1|rᵢ,ₚᵢ = 1|Lₘₜₜₓₜ,₀) in order to address the complexity problem. To make the constraint pᵢ = 1, we define the maximum process time and normalize with it the other problem variables in Definition 3.1. Let

\[ \theta_{\text{max}} := \max_{i \in C} \max_{x_i(0) \in X_i; y_i(0) = \alpha_i} \{t : y_i(t, u_i, M) = \beta_i\}, \]  

which determines the largest time required to cross the intersection among all controlled vehicles. Considering this as the fixed process time, we rewrite the relaxed version of Problem 3.2 as follows.

**Problem 4.1 (Relaxed Scheduling Problem).** Given an initial condition x(0), determine whether there exists a schedule \( T = (T_1, \ldots, T_{n_c}) \), where \( C = \{1, \ldots, n_c\} \), such that for all \( i \in C \),

\[ R_i \leq T_i \leq D_i \]  

for all \( i \neq j \in C \) and \( \gamma \in \bar{C} \) if \( T_i > 0 \),

\[ T_i \leq T_j \Rightarrow T_i + \theta_{\text{max}} \leq T_j \]  
\[ T_i \leq \bar{R}_\gamma \Rightarrow T_i + \theta_{\text{max}} \leq \bar{R}_\gamma \]  
\[ T_i \geq \bar{R}_\gamma \Rightarrow T_i \geq \bar{R}_\gamma \]  

if \( y_i(0) \geq \alpha_i \), i.e., \( T_i = 0 \), conditions (3.4) and (3.5) need to be satisfied instead of conditions (4.3) and (4.4).

Notice that (4.4) and (4.5) in Problem 4.1 imply that \( T_i \) for any \( i \in C \) cannot start during \( (\bar{R}_\gamma - \theta_{\text{max}}, \bar{P}_\gamma) \) for all \( \gamma \in \bar{C} \). Thus, we normalize these time intervals with \( \theta_{\text{max}} \) and let them be the initial forbidden region \( F \) in Algorithm 4.3. Assume that \( y_i(0) \geq \alpha_i \) for \( i \in \{1, \ldots, m\} \) and \( y_i(0) < \alpha_i \) for \( i \in \{m + 1, \ldots, n_c\} \).
Algorithm 4.4 Verification of Problem 4.1

procedure APPROXSOLUTION(x(0), δ)
    if y(0) ∈ B then return \{0, no\}
    for all i ∈ C and γ ∈ \(\bar{C}\) do
        given \(x_i(0)\) calculate \(R_i, D_i, \theta_{\text{max}}\)
        given \(x_\gamma(0)\) calculate \(\hat{R}_\gamma, \hat{P}_\gamma\)
        \(\hat{R}_\gamma \leftarrow \max(\hat{R}_\gamma - \delta, 0)\)
        \(\hat{P}_\gamma \leftarrow \max(\hat{P}_\gamma - \delta, 0)\)
    for all \(\gamma \in \bar{C}\) such that \(\hat{P}_\gamma > 0\) do
        \(F \leftarrow F \cup (\max(\hat{R}_\gamma/\theta_{\text{max}} - 1, 0), \hat{P}_\gamma/\theta_{\text{max}})\)
    for \(i \leftarrow 1\) to \(m\) do
        calculate \(P_i(T_i)\) and \(P_{\text{max}} \leftarrow \max_i P_i(T_i)\)
    for \(i \leftarrow m + 1\) to \(n\) do \(R_i \leftarrow \max(R_i, P_{\text{max}})\)
    \(r \leftarrow (R_{m+1}/\theta_{\text{max}}, \ldots, R_n/\theta_{\text{max}})\)
    \(d \leftarrow (D_{m+1}/\theta_{\text{max}} + 1, \ldots, D_n/\theta_{\text{max}} + 1)\)
    \([t_{m+1}, \ldots, t_n, \text{answer}] \leftarrow \text{POLYNOMIAL}(F, r, d)\)
    for \(i \leftarrow m + 1\) to \(n\) do \(T_i \leftarrow \theta_{\text{max}}t_i\)
    return \{T, \text{answer}\}

Algorithm 4.4 is the solution for Problem 4.1 when \(\delta = 0\).

4.1.2 Relaxed Verification Problem

Notice that Algorithm 4.4 may return \(\text{no}\) even when the solution for Problem 3.2 exists. We quantify how conservative Algorithm 4.4 is, which will be considered to formulate the Relaxed Verification Problem.

Lemma 4.1. Consider \(T_i\) and \(T_j\) for some \(i, j \in C\) with \(y_i(0) < \alpha_i\) and \(y_j(0) < \alpha_j\) and \(u_i\) and \(u_j\) such that \(y_i(T_i, u_i) = \alpha_i\) and \(y_j(T_j, u_j) = \alpha_j\). If \(T_i > T_j\) and \(T_i - T_j < \theta_{\text{max}}\) or if \((T_i, T_i + \theta_{\text{max}}) \cap (\hat{R}_\gamma, \hat{P}_\gamma) \neq \emptyset\) for some \(\gamma \in \bar{C}\), then there exists \(t^* \in [T_i, T_i + \theta_{\text{max}}]\) such that either \(y_j(t^*, u_j) \in (\alpha_j, \alpha_j + \theta_{\text{max}}\dot{y}_j, M)\) or \(y_\gamma(t^*, d_\gamma) \in (\alpha_\gamma, \beta_\gamma)\), respectively.

Proof. The condition \(T_i - T_j < \theta_{\text{max}}\) implies a violation of (4.3). Let \(t^* = T_i\). Since \(T_i > T_j\), we have \(y_j(t^*, u_j) > \alpha_j\). Also, the assumptions that \(T_i < T_j + \theta_{\text{max}}\) and \(\dot{y}_j \leq \dot{y}_j, M\) imply \(y_j(t^*, u_j) < \alpha_j + \theta_{\text{max}}\dot{y}_j, M\). Thus, \(t^*\) satisfies \(y_j(t^*, u_j) \in (\alpha_j, \alpha_j + \theta_{\text{max}}\dot{y}_j, M)\). The next condition \((T_i, T_i + \theta_{\text{max}}) \cap (\hat{R}_\gamma, \hat{P}_\gamma) \neq \emptyset\) implies that \(T_i\) does not
satisfy (4.4) and (4.5). That is, for an uncontrolled vehicle $\gamma$, there is $t^*$ such that $t^* \in (T_i, T_i + \theta_{\max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma)$. Since $\dot{y}_\gamma > 0$, for $t^* > \bar{R}_\gamma$ and $d_\gamma = d_{\gamma,m}$, we have $y_\gamma(t^*, d_\gamma) > \alpha_\gamma$; for $t^* < \bar{P}_\gamma$ and $d_\gamma = d_{\gamma,m}$, we have $y_\gamma(t^*, d_\gamma) < \beta_\gamma$. Therefore, for some $d_\gamma \in D_\gamma$, we have that $y_\gamma(t^*, d_\gamma) \in (\alpha_\gamma, \beta_\gamma)$. \hfill $\square$

Taking this lemma into account, we inflate the intersection with $\hat{\beta}_i := \alpha_i + \theta_{\max} \dot{y}_i,M$ for $i \in C$ and $\hat{\beta}_\gamma := \beta_\gamma$ for $\gamma \in \bar{C}$. Then, an inflated bad set is defined as follows:

$$\hat{B} := \{ y \in Y : y_i \in (\alpha_i, \hat{\beta}_i) \text{ and } y_j \in (\alpha_j, \hat{\beta}_j) \}$$

for some $i \neq j$ such that $i \in C$ and $j \in C \cup \bar{C}$. \hfill (4.6)

We then modify Problem 3.1 by substituting the inflated bad set $\hat{B}$ in place of the bad set $B$.

**Problem 4.2 (Relaxed Verification Problem).** *Given an initial condition $x(0)$, determine if there exists an input signal $u \in U$ which guarantees that $y(t, u, d) \notin \hat{B}$ for all $d \in D$ for all $t \geq 0$.*

**Theorem 4.1.** *Given $x(0)$ and $\delta = \{0, \tau\}$, if APPROXSOLUTION($x(0), \delta$) returns no, then there is no input $u$ such that $y(t, u, d) \notin \hat{B}$ for all $d$ for all $t \geq 0$.*

**Proof.** In Algorithm 4.4, APPROXSOLUTION($x(0), \delta$) returns no for two cases: $y(0) \in B$ or POLYNOMIAL($F, r, d$) returns no. In the first case, $y(0) \in \hat{B}$ because $B \subseteq \hat{B}$, so that at $t = 0$, $y(t, u, d) \in \hat{B}$ for any $u$. In the second case, according to Algorithm 4.3, POLYNOMIAL($F, r, d$) returns no when the conditions described in Lemma 4.1 hold. Thus, by Lemma 4.1, if APPROXSOLUTION($x(0), \delta$) returns no, then there exist some $d \in D$ and some $t \geq 0$ such that $y(t, u, d) \in \hat{B}$ for any $u$. Therefore, there is no input $u$ such that $y(t, u, d) \notin \hat{B}$ for all $d$ for all $t \geq 0$. \hfill $\square$

Theorem 4.1 indicates that if Algorithm 4.4 returns no, then there is no solution to Problem 4.2, thereby letting the efficient supervisor introduced in the next section intervene.
4.2 Efficient Supervisors

We now define an efficient supervisor \( \hat{s}(x(k\tau), a_k) \) by employing \( \hat{B} \) in place of \( B \) in Problem 3.3. Also, define the input operator for \( i \in C \) as follows:

\[
\hat{s}_i(x_i(0), T_i) := \arg \inf_{u_i \in U_i} \{ t \geq 0 : y_i(t, u_i) = \hat{\beta}_i \text{ with constraint } y_i(T_i, u_i) = \alpha_i \}. \quad (4.7)
\]

Moreover, define another efficient supervisor \( s_{\text{approx}}(x(k\tau), a_k) \) as in Algorithm 2 in which APPROX\textSOLUTION is used in place of EXACT\textSOLUTION and \( \hat{\sigma} \) is used in place of \( \sigma \). Here, we prove that \( \hat{s} \) is no more restrictive than \( s_{\text{approx}} \), that is, we prove that if \( \hat{s} \) is non-empty, \( s_{\text{approx}} \) is also non-empty.

Lemma 4.2. If \( \text{APPROX\textSOLUTION}(x(k\tau), 0) = \{T, yes\} \), then \( u := \hat{s}(x(k\tau), T) \neq \emptyset \). Furthermore, for \( u_k \) defined as \( u \) on \([k\tau, (k+1)\tau]\), \( \text{APPROX\textSOLUTION}(x_{k+1}(u_k), \tau) = \{T, yes\} \).

Proof. Same procedure as in proof of Lemma 3.1. \( \Box \)

Lemma 4.3. If \( \text{APPROX\textSOLUTION}(x_{k+1}(a_k), \tau) = \{T, yes\} \), then \( \hat{s}(x_{k+1}(a_k), T) \neq \emptyset \).

Proof. Same procedure as in proof of Lemma 3.2. \( \Box \)

Theorem 4.2. The supervisor \( s_{\text{approx}}(x(k\tau), a_k) \) is no more restrictive than \( \hat{s}(x(k\tau), a_k) \) and non-blocking.

Proof. If \( \hat{s}(x(k\tau), a_k) \) is non-empty, then \( \text{APPROX\textSOLUTION}(x_{k+1}(a_k), \tau) \) returns yes by Theorem 4.1 and definition of \( \hat{s} \). Then Lemma 4.3 implies that \( \hat{s}(x_{k+1}(a_k), T) \) is non-empty, which in turn implies that \( s_{\text{approx}}(x(k\tau), a_k) \) is non-empty by definition of \( s_{\text{approx}} \). Therefore, \( s_{\text{approx}}(x(k\tau), a_k) \) is no more restrictive than \( \hat{s}(x(k\tau), a_k) \). The non-blockingness follows the same procedure as in proof of Theorem 3.2. \( \Box \)

4.3 Simulation Results

The same parameters and dynamic model are considered as in Section 3.3. The initial condition used in this section is equivalent to the initial condition used in Figure 3-2
in Section 3.3.

Figure 4-1: Positions of 8 controlled (black solid line) and 2 uncontrolled (red dotted line) vehicles controlled by \( s_{approx} \) in Section 4.2. The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The darker shaded region represents the location of the intersection while the lighter shaded region represents that of the inflated intersection.

In Figure 4-1, Algorithm 3.2 using APPROXSOLUTION (Algorithm 4.4) in place of EXACTSOLUTION (Algorithm 3.1) is simulated to design \( s_{approx} \) for an example with 8 controlled and 2 uncontrolled vehicles. Figure 4-1 shows that the trajectories cross the darker shaded region without time overlapping, which implies the system's output avoids the bad set. Meanwhile, \( s_{approx} \) does not guarantee that only one vehicle stays inside the lighter shaded region at a time, that is, the trajectories are inside the inflated bad set. This, together with the result of Theorem 4.2, shows that \( s_{approx} \) is strictly less restrictive than \( \hat{s} \).

Recall that the simulation time per iteration for Figure 3-2 is 7.3 seconds on average. However, the simulation in Figure 4-1 takes 0.001 seconds per iteration on average and 0.07 seconds for the worst iteration. In light of the fact that 0.1 seconds is a usual practical sampling time, this algorithm is fast enough to be implemented in real-time applications.
Chapter 5

Optimal Approaches

This chapter designs two optimization problems on the top of the efficient solution, which is introduced in Chapter 4. By virtue of solving the optimization problems, new smoother inputs are achieved in the sense that the abrupt change of the inputs and unnecessary intervention are minimized. These problems must be solved in polynomial time to maintain fast computation time for real-time applications.

We specify the motivations of these optimization problems in Section 5.1 and formulate the problems called the Input Optimization Problem and the Schedule Optimization Problem in Section 5.2 and Section 5.3, respectively. The Input Optimization Problem is designed to replace the input operator $\hat{e}$ defined in (4.7), and the Schedule Optimization Problem is designed to optimize the solutions of Problem 4.1. Since we only override controlled vehicles with designed inputs, uncontrolled vehicles are not considered in this chapter.

5.1 Motivations

In this section, we present two issues of the control inputs that are generated by the Efficient Supervisor in Chapter 4. The first issue is the abrupt changes of inputs, which may cause inconvenience to drivers. The other issue is unnecessary intervention, which can be seen when there is one vehicle that is not involved in any collisions.
5.1.1 Abrupt Changes of Inputs

Consider an example with 4 controlled vehicles. The intersection is located at $[30m, 33m]$, and speed bounds are $[1m/s, 10m/s]$ with a sampling time $0.1sec$. The other parameters are equivalent to the example in Section 3.3. As been noticed in Chapter 4, one iteration takes at most $0.07sec$ using the efficient supervisor.

![Diagram of vehicle positions and supervisor overrides.](image)

Figure 5-1: Positions of 4 vehicles (black solid line). The blue boxes at the bottom represent the time steps when the supervisor overrides the controlled vehicles. The darker shaded region represents the location of the intersection while the lighter shaded region represents that of the inflated intersection.

Recall from Section 3.3 that the control input for this system is a longitudinal acceleration of the vehicles. The accelerations that we apply to vehicles usually show many oscillations as in Figure 5-2. Because of this flutter, the actuator could be damaged, and the car sways back and forth by inertia. Moreover, the drivers would feel the discrepancy between the input that they apply to the vehicles and the input that the supervisor imposes.

To address this issue and provide a more convenient driving environment to drivers, we propose the Input Optimization Problem in Section 5.2, where the difference between a control effort and a desired input is minimized for every step.
Figure 5-2: Control input (black solid line) for the last car with its desired input (blue dotted line). The two horizontal dotted lines represent the upper and lower bounds for the input.

5.1.2 Unnecessary Intervention

Suppose a different initial condition of 4 vehicles is given such that one vehicle travels far from an intersection compared to the other vehicles. All the parameters are the same as in Section 5.1.1. The Efficient Supervisor in Chapter 4 is used to control the vehicles.

The positions of the vehicles over time are shown in Figure 5-3(a). All of the vehicles are overridden when the blue boxes are present. However, in Figure 5-3(b), the last vehicle is not overridden and travels with its desired input. This still does not cause any collision because the last vehicle is not involved in any collision.
(a) All four vehicles are controlled by the Efficient Supervisor in Chapter 4

(b) The last vehicle is not in control

Figure 5-3: Positions of 4 vehicles where the last vehicle is far from the intersection compared to the others.
Figure 5-4 shows the input signal for the last vehicle in Figure 5-3(a). Although the intervention in the last vehicle is unnecessary as can be seen in Figure 5-3(b), the Efficient Supervisor overrides not only the other vehicles but also the last vehicle, which is not involved in collisions. In this individual perspective, the Efficient Supervisor is not least restrictive, that is, it violates that it intervenes only if necessary.

This issue mainly originates from the design of the decision version of scheduling problem as in Definition 2.1. Therefore, to address this problem, we propose optimizing a schedule to be closest to a desired schedule which is defined in Section 5.3.

## 5.2 Input Optimization Problem

To take advantage of a dynamic model in the design of an optimization problem, we consider a linear discrete system. Based on this model, we formulate the Discrete
Schedule Problem and the Input Optimization Problem.

5.2.1 Discrete Scheduling Problem

Recall the dynamic model of vehicle $i$:

\[
\dot{x}_i = f_i(x_i, u_i), \quad y_i = h_i(x_i). \tag{2.1}
\]

For a non-negative integer $k \in \mathbb{Z}_{\geq 0}$ where $\mathbb{Z}_{\geq 0}$ is a space of non-negative integers and for a sampling time $\Delta t$, let the dynamical state $x_i(k\Delta t)$ be denoted by $x_i[k]$ and the output $y_i(k\Delta t)$ by $y_i[k]$. The input $u_i(t)$ for $t \in [k\Delta t, (k + 1)\Delta t)$ is considered constant and denoted by $u_i[k]$. Through this chapter, we consider the corresponding linear discrete time-invariant system to (2.1) denoted as follows:

\[
x_i[k + 1] = A_i x_i[k] + B_i u_i[k], \quad y_i[k] = C_i x_i[k], \tag{5.1}
\]

where $A_i$ is the system matrix, $B_i$ is the input matrix, and $C_i$ is the output matrix for each vehicle $i$. The input signal for $t \in [0, (N + 1)\Delta t)$ is a vector denoted by $u_i := (u_i[0], \ldots, u_i[N])$. Similar to the continuous system, $\hat{y}_i[k] \in [\hat{y}_{i,m}, \hat{y}_{i,M}]$ and $u_i[k] \in [u_{i,m}, u_{i,M}]$ for all $k \in \mathbb{Z}_{\geq 0}$. The output trajectory $y_i[k, u_i, x_i[k_0]]$ and state $x_i[k, u_i, x_i[k_0]]$ denote the output and state reached at $t = (k + k_0)\Delta t$ with an input signal $u_i$ starting from $x_i[k_0]$, respectively. We assume that both $y_i[k, u_i, x_i[k_0]]$ and $x_i[k, u_i, x_i[k_0]]$ are order preserving functions of their arguments. For notational brevity, we omit an initial condition when $k_0 = 0$. The aggregate state, output, and input are denoted by $x[k], y[k]$, and $u[k]$, respectively.

Based on the dynamic model (5.1), we define the variables of discrete scheduling. To ensure the existence of these variables, a truncation error $\Delta \alpha_i$ is considered.

**Definition 5.1.** For all $i$ where $y_i[0] < \alpha_i - \Delta \alpha_i$, let

\[
R_i := \min_{u_i} \{ k \in \mathbb{Z}_{\geq 0} : \alpha_i - \Delta \alpha_i \leq y_i[k, u_i] \leq \alpha_i \},
\]
Given a non-negative integer $T_i \in \mathbb{Z}_{\geq 0}$, 

$$
P_i(T_i) := T_i + \vartheta_{\text{max}},
$$

where 

$$
\vartheta_{\text{max}} := \max \min \{k \in \mathbb{Z}_{\geq 0} : y_i[k, u_i] \geq \beta_i\}.
$$

If $\alpha_i - \Delta \alpha_i \leq y_i[0] < \alpha_i$, let $R_i := 0$, and $D_i := \max_{u_i} \{k \in \mathbb{Z}_{\geq 0} : \alpha_i - \Delta \alpha_i \leq y_i[k, u_i] \leq \alpha_i\}$. If $\alpha_i \leq y_i[0] < \beta_i$, let $R_i := 0$ and $D_i := 0$. For both cases, that is, $\alpha_i - \Delta \alpha_i \leq y_i[0] < \beta_i$, given a non-negative integer $T_i \in \mathbb{Z}_{\geq 0}$, 

$$
P_i(T_i) := \min_{u_i} \{k \in \mathbb{Z}_{\geq 0} : y_i[k, u_i] \geq \beta_i \text{ with constraint } y_i[T_i] \leq \alpha_i\}.
$$

If the constraint $y_i[T_i, u_i] \leq \alpha_i$ is not satisfied, set $P_i(T_i) = \infty$. If $y_i[0] \geq \beta_i$, then set $R_i = 0, D_i = 0$, and $P_i(T_i) = 0$.

We now formulate the Discrete Scheduling Problem. This problem requires the same constraints as Problem 5.1 except the idle-time constraints. We consider $n$ vehicles approaching an intersection.

**Problem 5.1 (Discrete Scheduling Problem).** *Given an initial condition $x[0]$, determine whether there exists a schedule $T = (T_1, \ldots, T_n)$ where $T_i \in \mathbb{Z}_{\geq 0}$ for all $i$ such that for all $i \in \{1, \ldots, n\}$, 

$$
R_i \leq T_i \leq D_i,
$$

for all $i \neq j$, 

$$
T_i \leq T_j \Rightarrow P_i(T_i) \leq T_j.
$$

Notice that a truncation error $\Delta \alpha_i$ appears when $R_i$ and $D_i$ are defined. This implies that a feasible schedule $T_i$ obtained from Problem 5.1 satisfies 

$$
\alpha_i - \Delta \alpha_i \leq y_i[T_i] \leq \alpha_i.
$$
Similarly, from the definition of $P_i(T_i)$, a truncation error $\Delta \beta_i$ can be introduced where $\Delta \beta_i$ satisfies

$$\beta_i \leq y_i[P_i(T_i), u_i] \leq \hat{\beta}_i + \Delta \beta_i.$$  

Notice that we consider $\hat{\beta}_i$ to take the definition of $\theta_{\text{max}}$ into account, where $\hat{\beta}_i := \alpha_i + \theta_{\text{max}} y_{i,M}$ defined from a result of Lemma 4.1.

Therefore, the inflated bad set from discretization is defined as follows:

$$\hat{B} := \{y \in Y : \exists t \geq 0 \text{ such that } y_i(t) \in (\alpha_i - \Delta \alpha_i, \hat{\beta}_i + \Delta \beta_i) \text{ and } y_j(t) \in (\alpha_j - \Delta \alpha_j, \hat{\beta}_j + \Delta \beta_j) \text{ for some } i \neq j\}. \quad (5.2)$$

where $\Delta \alpha_i \geq 0$ and $\Delta \beta_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Notice that $\hat{B}$ considers the continuous dynamics while Problem 5.1 considers the discrete dynamics.

Similar to Problem 4.2, we then modify Problem 3.1 by substituting the inflated bad set from discretization $\hat{B}$ in place of the bad set $B$ and eliminating the disturbance of uncontrolled vehicles.

**Problem 5.2** (Relaxed Verification Problem for Discretization). *Given an initial condition* $x(0)$, *determine if there exists an input signal* $u \in U$ *which guarantees that* $y(t, u) \notin \hat{B}$ *for all* $t \geq 0$.

The relation between this verification problem (Problem 5.2) and the Discrete Scheduling Problem (Problem 5.1) is that if Problem 5.1 returns *no*, that Problem 5.2 always returns *no*. The converse is not true. Moreover, if Problem 5.1 returns *yes*, then there exists an input to avoid the original bad set $B$, that is, Problem 3.1 returns *yes*. These are proved in Appendix A.2. Thus, we design a discrete supervisor such that it overrides the system when Problem 5.1 returns *no*.

### 5.2.2 Input Optimization Problem

In the design of a discrete supervisor, the most important part is to define an input generator such as $\sigma$ in (3.8) and $\hat{\sigma}$ in (4.7) because the safety is directly guaranteed from an input operator. While $\sigma$ and $\hat{\sigma}$ randomly select the input that satisfies their
definitions, in this section an input operator is defined to minimize the difference between a control input and a desired input by a driver, as mentioned in Section 5.1.1.

To implement an optimization problem, we use model predictive control (MPC), which is widely used for the on-line computations of optimization problems with constraints. The MPC encompasses open-loop feedback control and certainty equivalence [7]. The open-loop feedback control is a classical suboptimal scheme that selects the present control input at each period \( \tau \) as if future measurements will be disregarded while indeed it updates the measurements in future step. When a disturbance is applied to a system, the computation is simplified via certainty equivalence, which at each period assumes a fixed disturbance at a nominal value.

In the formulation of the Input Optimization Problem, a desired input plays a role of a disturbance, which can be regarded constant as in the certainty equivalence scheme. Let a desired input vector \( a_{k,i} \) be a constant vector of a desired input \( a_{k,i} \) in a \( (P_i(T_i) - 1) \)-dimensional space for a given process time \( P_i(T_i) \). Also, we assume the time step for the supervisor \( \tau \) is a multiple of the sampling time for the discrete dynamic \( \Delta t \), that is, \( \tau = m \cdot \Delta t \) where \( m \in \mathbb{Z}_{\geq 0} \). The weighting matrix \( Q \) is a symmetric positive semi-definite and diagonal matrix. Since the MPC selects the present control input, we choose \( Q = \text{diag}([1,0,...,0]) \) to weight only the present input.

**Problem 5.3 (Input Optimization Problem).** For each vehicle \( i \) at time step \( K \) and for given a schedule \( T_i \) and a process time \( P_i(T_i) \), find \( u_i \) such that

\[
\min_{u_i} (u_i - a_{K,i})'Q(u_i - a_{K,i})
\]

subject to

for all \( k \in \{0, \ldots, P_i(T_i) - 1\} \),

\[
\dot{y}_{i,m} \leq \dot{y}_i[k] \leq \dot{y}_{i,M},
\]

\[
u_{i,m} \leq u_i[k] \leq u_{i,M},
\]

55
\[ \alpha_i - \Delta \alpha_i \leq y_i[T_i, u_i] \leq \alpha_i, \]
\[ y_i[P_i(T_i), u_i] \geq \beta_i. \]

Notice that due to the constraints, the solution of Problem 5.3 guarantees safety if \([T_i, P_i(T_i)]\) does not overlap for all \(i\). Indeed, Problem 5.1 finds such a schedule. Provided that the weighing matrix \(Q\) is positive semi-definite, Problem 5.3 is a convex quadratic optimization problem, which can be easily solved.

The existence of a feasible input for Problem 5.3 is presented in Appendix A.1.

### 5.2.3 Simulation Results

The input vector solving Problem 5.3 minimizes the cost at every step. However, this input vector and the actual control input are different due to the change from future measurements, the uncertainties from a disturbance, and the construction of the supervisor. To visualize the improvement achieved by solving the Input Optimization Problem, we generate 60 different initial conditions for 4 vehicles and compute the square of the difference between the actual control input and the desired input for each vehicle. For each sample of initial conditions, we find a stage \(P_i\) when vehicle \(i\) exits an intersection and then compute the average cost,

\[
\frac{\sum_{k=0}^{P_i-1} (u_i[k] - a_{k,i})^2}{P_i}, \tag{5.3}
\]

for each vehicle. Thus, each sample contains 4 average costs for each vehicle. Finally, we sort these 4 costs in an increasing order and call the one that has the minimum cost by the best performance and the one that has the maximum cost by the worst performance. The following figures show the average cost across 60 samples from the best to worst performance.

In Figure 5-5, we observe that the worst performance deteriorates in the optimal
control that is obtained from Problem 5.3 while the others improve in the optimal control input.

5.3 Schedule Optimization Problem

Consider a vehicle $i$ travelling far from an intersection while the others are not, as in Section 5.1.2. Recall that the Relaxed Scheduling Problem (Problem 4.1) is a decision problem, which returns a binary answer. Thus, Algorithm 4.4 searches the soonest possible schedule, more precisely, the schedule that minimizes $L_{max}$, and checks if the schedule violates $T_i \leq D_i$. Because of this soonest schedule scheme, vehicle $i$ is assigned the soonest possible schedule at each stage, thereby being forced to accelerate when collisions among the others become urgent.

To address this issue of unnecessary intervention, we formulate another optimization problem. Since in this chapter, we consider the discrete dynamic model and thus an integer schedule, optimizing a schedule becomes an integer linear optimization
problem. To solve this problem in polynomial time, we introduce a special class of matrix in Section 5.3.1 and transform the integer linear optimization problem into the equivalent linear optimization problem in Section 5.3.2. In Section 5.3.3, simulation results are given.

5.3.1 Totally Unimodular Matrix [20]

Since a schedule obtained from the Discrete Schedule Problem is integer, the Schedule Optimization Problem becomes an integer linear optimization problem, which can be formulated as follows:

$$\min\{c'x : Ax \leq b, x \text{ integers}\}. \quad (5.4)$$

Because of the integer constraint, this problem is known to be NP-hard. However, for a special class of $A, c$ and $b$, this problem can be transformed to a linear optimization problem. The same problem without the integer constraints, that is, $\min\{c'x : Ax \leq b\}$, is called the LP-relaxation of the problem (5.4).

Consider a feasible set $\{x : Ax \leq b\}$. If every element of a matrix $A$ is integer, $A$ is called an integral matrix. If every element of a vector $b$ is integer, $b$ is called an integral vector. A square integral matrix $B$ is "unimodular" if $\det B = \pm 1$. An integral matrix $A$ is "totally unimodular" if every square non-singular submatrix of $A$ is unimodular.

**Proposition 5.1 (p. 269).** Let $A$ be a totally unimodular matrix, and let $b$ be an integral vector. Then, the polyhedron $\{x : Ax \leq b\}$ has only integral vertices.

**Proposition 5.2 (p. 267).** If $A$ is a totally unimodular matrix, and $b$ and $c$ are integral vectors, then the problem

$$\min\{c'x : Ax \leq b\}$$

has only integral optimum solutions.

Proposition 5.2 implies that given a totally unimodular matrix $A$ and integral vectors $b$ and $c$, the LP relaxation of (5.4) retains only integral solutions, which
satisfy the integer constraint in (5.4). That is, the solutions for the LP relaxation become the solutions for the original problem. Since a linear optimization problem can be solved in polynomial time, the corresponding integer linear optimization problem can be solved in polynomial time.

5.3.2 Schedule Optimization Problem

As we define a desired input vector \( a_{K,i} \) for vehicle \( i \) at stage \( K \) as a constant vector to simplify computation, we define a desired schedule \( T_{d,i} \) for \( y_i[0] < \alpha_i - \Delta \alpha_i \) as follows:

\[
T_{d,i} := \{ k \in \mathbb{Z}_{\geq 0} : \alpha_i - \Delta \alpha_i \leq y_i[k, a_{K,i}] \leq \alpha_i \}. 
\]

(5.5)

The goal of this section is to minimize the difference between an actual schedule \( T \) and a desired schedule \( T_d \), where \( T_d \) is an aggregate vector of \( T_{d,i} \). By solving such an optimization problem, the schedule \( T_i \) for vehicle \( i \) is equivalent to \( T_{d,i} \) if vehicle \( i \) does not have to change its input, thereby, minimizing the occurrence of unnecessary intervention. For the Schedule Optimization Problem, we only consider the vehicles that satisfy \( y_i[0] < \alpha_i - \Delta \alpha_i \).

Let \( \pi \) be a given sequence of \( n \) vehicles. This sequence can be obtained from Algorithm 4.4. If Algorithm 4.4 returns yes, we take the feasible sequence; otherwise, there is no feasible schedule so that we do not have to solve a further optimization problem. The \( i \)th element of \( \pi \) is denoted by \( \pi_i \).

**Problem 5.4 (Schedule Optimization Problem).** Find a schedule \( T = (T_1, \ldots, T_n) \) such that

\[
\min_{\pi} \sum_{i=1}^{n} |T_i - T_{d,i}|
\]

subject to

\[
R_i \leq T_i \leq D_i, \quad \forall i \in \{1, \ldots, n\},
\]

\[
P_{\pi_i}(T_{\pi_i}) \leq T_{\pi_j}, \quad \forall i < j,
\]

\[
T_i \in \mathbb{Z}_{\geq 0}, \quad \forall i.
\]

Notice that via the constraints, a feasible schedule must solve the Discrete Scheduling Problem (Problem 5.1). That is, Problem 5.4 chooses the best schedule in terms
of the objective function among all possible schedules that satisfy Problem 5.1. Since all vehicles of interest satisfy $y_i[0] < \alpha_i - \Delta \alpha_i$, we have $P_i(T_i) = T_i + \vartheta_{\text{max}}$.

In general, the absolute objective function can not be solved in polynomial time [looking for a reference]. To modify the objective function into a solvable one, define non-negative integers $T_i^+$ and $T_i^-$ as follows:

$$T_i^+ := \max((T_i - T_{d,i}), 0) \quad \text{and} \quad T_i^- := \max(0, -(T_i - T_{d,i})).$$

Then they also satisfy $T_i - T_{d,i} = T_i^+ - T_i^-$ while minimizing $|T_i - T_{d,i}| = T_i^+ + T_i^-$. Let $T_+ := (T_{\pi_1}^+, \ldots, T_{\pi_n}^+)$ and $T_- := (T_{\pi_1}^-, \ldots, T_{\pi_n}^-)$. The aggregate column vector in $Z^{2n}$ is denoted by $T^\pm := (T^+, T^-)'$. Now we reformulate Problem 5.4 into a linear programming format.

**Problem 5.5** (Reformulation of Problem 5.4). Find a vector $T^\pm = (T^+, T^-)'$ such that

$$\min_{T_i^+, T_i^-} \sum_{i=1}^{n}(T_i^+ + T_i^-)$$

subject to

$$R_i \leq (T_{\pi_i}^+ - T_{\pi_i}^-) + T_{d,i} \leq D_i, \quad \forall i$$

$$(T_{\pi_i}^+ - T_{\pi_i}^-) + T_{d,\pi_i} + \vartheta_{\text{max}} \leq (T_{\pi_j}^+ - T_{\pi_j}^-) + T_{d,\pi_j}, \quad \forall i < j,$$

$$T_i^+, T_i^- \in Z_{\geq 0}, \quad \forall i.$$

By the definition of the LP relaxation of an linear integer programming, given in Section 5.3.1, the LP relaxation of Problem 5.5 is as follows.

**Problem 5.6** (LP relaxation of Problem 5.5). Find a vector $T^\pm = (T^+, T^-)'$ such that

$$\min_{T_i^+, T_i^-} \sum_{i=1}^{n}(T_i^+ + T_i^-)$$

60
subject to

\[
\begin{align*}
    R_i &\leq (T_i^+ - T_i^-) + T_{d,i} \leq D_i, \quad \forall i \\
    (T_{\sigma_i}^+ - T_{\sigma_i}^-) + T_{d,\sigma_i} + \vartheta_{\text{max}} &\leq (T_{\sigma_j}^+ - T_{\sigma_j}^-) + T_{d,\sigma_j}, \quad \forall i < j, \\
    T_i^+ &\geq 0, T_i^- \geq 0, \quad \forall i.
\end{align*}
\]

Problem 5.6 can be rewritten as \( \min \{c'T^\pm : AT^\pm \leq b\} \), where the explicit expressions of \( A, b \), and \( c \) are given in Appendix B (under construction...). Given the fact that the variables \( R_i, D_i, T_{d,i} \), and \( \vartheta_{\text{max}} \) are integers, it can be verified also in Appendix B that \( A \) is a totally modular matrix, and \( b \) and \( c \) are integral vectors. By Proposition 5.2, Problem 5.6 has only integral optimum solutions. These solutions also satisfy the integer constraints in Problem 5.5. That is, the solutions for Problem 5.6 become the solutions for Problem 5.5.

Using these optimal solutions for Problem 5.6 denoted by \( T_i^{++} \) and \( T_i^{--} \) for all \( i \), the solutions for Problem 5.4 can be obtained by \( T_i^* = (T_i^{++} - T_i^{--}) + T_{d,i} \).

### 5.3.3 Simulation Results

In this section, we implement the optimal inputs derived from the Input Optimization Problem and the Schedule Optimization Problem, as described in Figure 5-6.
Algorithm 4.4 If answer = yes, Input Optimization Problem.

If answer = no, Store \( u_{k+1, safe} \).

Figure 5-6: Implementation of the two optimization problems (Problem 5.3 and Problem 5.4) at stage \( k \). First, the Discrete Scheduling Problem is solved via Algorithm 4.4, and if there exists a feasible schedule, the sequence of the schedule lets the Schedule Optimization Problem compute the optimal schedule \( T^* \). If there does not exist a feasible schedule, Algorithm 4.4 returns no and an empty schedule. Depending on this answer, the supervisor either runs the Input Optimization Problem to store the safe input for the next step \( u_{k+1, safe} \) or overrides the system using the safe input \( u_{k, safe} \).

In the first example, the same initial condition and desired input as in Figure 5-3 in Section 5.1.2 are considered. By solving the Input Optimization Problem and the Schedule Optimization Problem at each period \( \tau \), the unnecessary intervention that used to be observed in Figure 5-4 does not appear as can be seen in Figure 5-7(b). However, although all the problems can be solved in polynomial time, this simulation takes 0.13 sec per iteration on average and 2.46 sec on the worst case. A larger sampling time for the supervisor \( \tau \) than the computation time can tackle this problem but make the supervisor conservative.
To confirm that this optimal scheme also works with a general desired input, a varying desired input is considered in the next example.

Recall that the definition of a desired schedule $\mathcal{T}_{d,i}$ in (5.5) is defined with a constant desired input because the future desired input is unknown. As can be seen in Figure 5-8(b), unnecessary intervention for the last vehicle, which is not involved
in a collision, can be prevented even for an unpredicted varying desired input. Also it is noteworthy that at 4sec, the supervisor intervenes, as known from the presence of the blue box in Figure 5-8(a), and there is one step delay for the control input to follow the change of the desired input. This is because the change of the desired input is unknown at the time when the control input is generated.

To provide a global quantification, we generate 60 random initial conditions of 4 vehicles, where one vehicle starts between 50 m and 60 m before an intersection, and the others start between 20 m and 30 m before the intersection. That is, one
vehicle travels farther from an intersection than the others. Then, using the metric of the difference between a control input and a desired input in (5.3), we compare the inputs obtained by solving the Input Optimization Problem and the inputs obtained by solving both the Input Optimization Problem (Problem 5.3) and the Schedule Optimization Problem (Problem 5.4).

Figure 5-9: Average cost on 60 samples of four vehicles. Random schedule (blue bars) represents the results from solving Problem 5.3 while Optimal schedule (red bars) represents the results from solving Problem 5.3 and Problem 5.4.

In Figure 5-9, the results present that solving Problem 5.4 along with Problem 5.3 generates smoother control inputs for most of vehicles than solving only Problem 5.4.
Chapter 6

Conclusions and Future Works

In this paper, we have designed a supervisor to prevent collisions among multiple heterogeneous vehicles. The design of the supervisor is based on two main problems: verification of the system safety, and construction of a safe input for the controlled vehicles. An inserted idle-time (IIT) scheduling is employed to determine safety in the presence of uncontrolled vehicles. The supervisor is least restrictive by construction, that is, it intervenes only if necessary.

Since the exact problem is NP-hard, we also introduced a relaxed problem and used an efficient scheduling algorithm with polynomial complexity to solve it. A quantification of conservatism of the approximate solution was also provided. To mitigate the abrupt changes of control inputs and to reduce the number of unnecessary interventions, additional optimization problems were formulated, which can be solved in polynomial time. The simulation results showed that the computation time exceeded 0.1 sec. Thus, to apply the problems to real-time applications, we need to enlarge the step size of the supervisor.

While the extension that we have presented in this paper to consider uncontrolled vehicles and to generate smoother control inputs makes the scheduling approach more practical, a number of real-world issues still need to be tackled. These include considering also rear-end collisions, the fact that real intersections include multiple conflict points, that a vehicle's path may be not known a priori, and that vehicle positions and speeds may be subject to measurement uncertainty. The issue of the multiple
conflict points, which relaxes the assumption of the intersection model in Section 2.2, is currently being investigated.
Appendix A

Proof for the discrete system

A.1 Existence of input

Proposition A.1. If there exists a schedule $T_i \in [R_i, D_i] \subset \mathbb{Z}_{\geq 0}$, then there exists an input vector $u^T_i := (u_i(0), \ldots, u_i(T_i - 1))$ such that $u_i[k] \in [u_{i,m}, u_{i,M}]$ for all $k = \{0, \ldots, T_i - 1\}$ and $\alpha_i - \Delta \alpha_i \leq y_i[T_i, u^T_i] \leq \alpha_i$.

Proof. Consider an input $u_i[0], \ldots, u_i[N - 1]$, where $N \in \mathbb{Z}_{\geq 0}$. From the discrete system dynamics (5.1), the future state $x_i[N]$ for vehicle $i$ at stage $N$ can be written for a given initial condition $x_i[0] \in \mathbb{R}^n$ as follows:

$$x_i[1] = A_ix_i[0] + B_iu_i[0],$$

$$x_i[2] = A_ix_i[1] + B_iu_i[1] = A^2_ix_i[0] + A_iB_iu_i[0] + B_iu_i[1],$$

$$\vdots$$

$$x_i[N] = A^N_i x_i[0] + A_i ^{N-1}B_iu_i[0] + \ldots + B_iu_i[N - 1],$$

$$= A^N_i x_i[0] + \left[ A_i ^{N-1}B_i, \ldots, B_i \right] \begin{pmatrix} u_i[0] \\ \vdots \\ u_i[N-1] \end{pmatrix},$$

$$:= A^N_i x_i[0] + \Psi^N_i u^N_i,$$

where we define a matrix $\Psi^N_i$ as $[A_i ^{N-1}B_i, A_i ^{N-2}B_i, \ldots, B_i]$ and a vector $u^N_i$ as $(u_i[0], \ldots, u_i[N-1])$. 


Thus, the output at stage $N$ becomes

$$y_i[N] = C_iA_i^N x_i[0] + C_i\Psi_i^N u_i^N.$$ 

Recall that for an input vector $u_i^N$, the output $y_i[N, u_i, x_i[0]]$ is an order preserving function with respect to the arguments. That is, for $N_1 \leq N_2 \in \mathbb{Z}_{\geq 0}$, we have $y_i[N_1, u_i^{N_1}, x_i[0]] \leq y_i[N_2, u_i^{N_2}, x_i[0]]$, where $u_i^{N_2}$ contains the same elements of $u_i^{N_1}$ from 1st to $N_1$th elements and is augmented by $(N_2 - N_1)$ random inputs. By choosing $u_i^{N_1} = 0$ and $u_i^{N_2} = 0$, we have

$$C_iA_i^{N_1} x_i[0] \leq C_iA_i^{N_2} x_i[0]. \quad (A.1)$$

Furthermore, by choosing $x_i[0] = 0$, we have $C_i\Psi_i^{N_1} u_i^{N_1} \leq C_i\Psi_i^{N_2} u_i^{N_2}$. By letting $C_i\Psi_i^N = (c_0, \ldots, c_{N-1})$, we have

$$\sum_{k=0}^{k=N_1-1} c_k u_i[k] \leq \sum_{k=0}^{k=N_2-1} c_k u_i[k],$$

which becomes $0 \leq \sum_{k=N_1}^{N_2-1} c_k u_i[k]$ for any $u_i[k]$. This implies that $c_k$ for all $k$ is non-negative.

By the definitions of $\mathcal{R}_i$ and $\mathcal{D}_i$ in Definition 5.1, there must exist input vectors $u_i^{\mathcal{R}_i}$ and $u_i^{\mathcal{D}_i}$ that satisfy $\alpha_i - \Delta \alpha_i \leq y_i[\mathcal{R}_i, u_i^{\mathcal{R}_i}] \leq \alpha_i$ and $\alpha_i - \Delta \alpha_i \leq y_i[\mathcal{D}_i, u_i^{\mathcal{D}_i}] \leq \alpha_i$, respectively. For $N = \mathcal{R}_i$,

$$\alpha_i - \Delta \alpha_i \leq y_i[\mathcal{R}_i, u_i^{\mathcal{R}_i}] \leq \alpha_i,$$

$$\alpha_i - \Delta \alpha_i \leq C_iA_i^{\mathcal{R}_i} x_i[0] + C_i\Psi_i^{\mathcal{R}_i} u_i^{\mathcal{R}_i} \leq \alpha_i,$$

$$\alpha_i - \Delta \alpha_i - C_iA_i^{\mathcal{R}_i} x_i[0] \leq C_i\Psi_i^{\mathcal{R}_i} u_i^{\mathcal{R}_i} \leq \alpha_i - C_iA_i^{\mathcal{R}_i} x_i[0].$$

Notice that $C_i\Psi_i^{\mathcal{R}_i}$ is a vector so that let $C_i\Psi_i^{\mathcal{R}_i} = (c_0, \ldots, c_{\mathcal{R}_i-1})$.

Let the 1-norm $\|C_i\Psi_i^{\mathcal{R}_i}\|_1$ represent $|c_0| + \ldots + |c_{\mathcal{R}_i-1}| = c_0 + \ldots + c_{\mathcal{R}_i-1}$ because $c_k$ for all $k \in \{0, \ldots, \mathcal{R}_i - 1\}$ are non-negative, and let $u_i^{\mathcal{R}_i} = (u[0], \ldots, u[\mathcal{R}_i - 1])$ in
an abuse of notation. Then,

\[
C_i \Psi_i^{R_i} u_i^{R_i} = c_0 u[0] + \ldots + c_{R_i-1} u[R_i - 1]
\]

\[
= \| C_i \Psi_i^{R_i} \|_1 \left( \frac{c_0}{\| C_i \Psi_i^{R_i} \|_1} u[0] + \ldots + \frac{c_{R_i-1}}{\| C_i \Psi_i^{R_i} \|_1} u[R_i - 1] \right)
\]

\[
= \| C_i \Psi_i^{R_i} \|_1 u_{R_i}.
\]

The last equation comes from the mean-value theorem. Since all elements of \( u_i^{R_i} \) are inside \( [u_{i,m}, u_{i,M}] \), we have \( u_{R_i} \in [u_{i,m}, u_{i,M}] \subset \mathbb{R} \).

Thus, for \( R_i \) and \( D_i \),

\[
\frac{\alpha_i - \Delta \alpha_i - C_i A_i^{R_i} x_i[0]}{\| C_i \Psi_i^{R_i} \|_1} \leq u_{R_i} \leq \frac{\alpha_i - C_i A_i^{R_i} x_i[0]}{\| C_i \Psi_i^{R_i} \|_1}, \quad (A.2)
\]

\[
\frac{\alpha_i - \Delta \alpha_i - C_i A_i^{D_i} x_i[0]}{\| C_i \Psi_i^{D_i} \|_1} \leq u_{D_i} \leq \frac{\alpha_i - C_i A_i^{D_i} x_i[0]}{\| C_i \Psi_i^{D_i} \|_1}. \quad (A.3)
\]

Let \( u_{T_i} \in \mathbb{R} \) be a corresponding input to \( u_i^{T_i} \) such that \( \alpha_i - \Delta \alpha_i \leq y_i[T_i, u_i^{T_i}] \leq \alpha_i \), and then,

\[
\frac{\alpha_i - \Delta \alpha_i - C_i A_i^{T_i} x_i[0]}{\| C_i \Psi_i^{T_i} \|_1} \leq u_{T_i} \leq \frac{\alpha_i - C_i A_i^{T_i} x_i[0]}{\| C_i \Psi_i^{T_i} \|_1}. \quad (A.4)
\]

For the existence of such \( u_i^{T_i} \), we must show that \( u_{T_i} \) is inside \( [u_{i,m}, u_{i,M}] \). In other words, the proof completes if we can find any \( u_{T_i} \) that satisfy \( u_{T_i} \in [u_{i,m}, u_{i,M}] \) and (A.4).

We prove this by contradiction. Suppose any \( u_{T_i} \) satisfying (A.4) is not feasible, that is, \( u_{T_i} < u_{i,m} \) or \( u_{T_i} > u_{i,M} \).

If \( u_{T_i} < u_{i,m} \), then from (A.4),

\[
\frac{\alpha_i - C_i A_i^{T_i} x_i[0]}{\| C_i \Psi_i^{T_i} \|_1} < u_{i,m}.
\]

Since \( D_i \geq T_i \), we have \( \| C_i \Psi_i^{D_i} \|_1 \geq \| C_i \Psi_i^{T_i} \|_1 \) because each element of \( C_i \Psi_i^{D_i} \) is
non-negative and $C_iA_i^x_i[0] \geq C_iA_i^{T_i}x_i[0]$ by (A.1). Using (A.3), we have

$$u_{D_i} \leq \frac{\alpha_i - C_iA_i^P_i x_i[0]}{||C_i\Psi_i^P||_1} \leq \frac{\alpha_i - C_iA_i^{T_i}x_i[0]}{||C_i\Psi_i^{T_i}||_1} < u_{i,m}.$$

This contradicts the fact that $u_{D_i} \in [u_{i,m}, u_{i,M}]$. The same procedure for the case of $u_{T_i} > u_{i,M}$ leads to $u_{R_i} > u_{i,M}$, which contradicts the fact that $u_{R_i} \in [u_{i,m}, u_{i,M}]$.

Therefore, there must exist $u_{T_i}$ corresponding to $u_{T_i} \in [u_{i,m}, u_{i,M}]$ such that $\alpha_i - \Delta \alpha_i \leq y_i(T_i, u_{T_i}) \leq \alpha_i$. \hfill $\square$

By the definition of $P_i(T_i)$ in Definition 5.1, $P_i(T_i) := \min_{u_i} \{ k \in \mathbb{Z}_{\geq 0} : y_i[k, u_i] \geq \beta_i \}$ with constraint $y_i[T_i] \leq \alpha_i$, we can obtain the input $u_{T_i}^{P_i}$ such that $y_i[P_i(T_i), u_{T_i}^{P_i}] \geq \beta_i$ and $y_i[T_i, u_{T_i}^{P_i}] \leq \alpha_i$.

### A.2 Relation between Discrete Scheduling Problem and Verification Problem

In Chapter 5, the Discrete Scheduling Problem (Problem 5.1) and the corresponding Relaxed Verification Problem (Problem 5.2) are formulated. In this section, we also consider the Verification Problem (Problem 3.1) to see what the answers of the Discrete Scheduling Problem imply.

**Proposition A.2.** If there is no schedule for Problem 5.1, neither is an input $u(t)$ for Problem 5.2.

**Proof.** The condition implies that there exist $T_i \in [R_i, D_i]$ and $T_j \in [R_j, D_j]$ for $i \neq j$ such that satisfy $T_i \leq T_j$ and $P_i(T_i) > T_j$, which violates the constraint $P_i(T_i) \leq T_j$. From Proposition A.1, there are input vectors $u_i$ and $u_j$ that satisfy $\alpha_i - \Delta \alpha_i \leq y_i[T_i, u_i] \leq \alpha_i$ and $\alpha_j - \Delta \alpha_j \leq y_j[T_j, u_j] \leq \alpha_j$, respectively. Since $T_i, T_j$ and $P_i(T_i)$ are integers, we have

$$T_i \leq T_j < T_j + 1 \leq P_i(T_i).$$

72
Now consider the continuous outputs and prove that they are inside the inflated bad set from discretization $ \hat{B} $ in (5.2). The corresponding continuous input $ u_i(t) $ to $ u_i $ is a piecewise constant with $ u_i(t) = u_i[k] $ for $ t \in [k\Delta t, (k + 1)\Delta t) $. Similarly, the continuous input $ u_j(t) $ corresponds to $ u_j $. At $ t^* = (T_j + \frac{1}{2})\Delta t $,

$$ \alpha_j - \Delta \alpha_j + \frac{1}{2} \dot{y}_{j,m} \Delta t \leq y_j(t^*, u_j(t)) \leq \alpha_j + \frac{1}{2} \dot{y}_{j,m} \Delta t. $$

Since $ (T_i + \frac{1}{2})\Delta t \leq t^* \leq (P_i(T_i) - \frac{1}{2})\Delta t $,

$$ \alpha_i - \Delta \alpha_i + \frac{1}{2} \dot{y}_{i,m} \Delta t \leq y_i(t^*, u_i(t)) \leq \hat{\beta}_i + \Delta \beta_i - \frac{1}{2} \dot{y}_{i,m} \Delta t. $$

These inequalities hold because the output function $ y_i(t) $ is non-decreasing in time. Assume that for all $ i $, $ \Delta \beta_i $ are greater than $ \frac{1}{2} \dot{y}_{i,m} \Delta t $. Then, $ y_j(t^*, u_j(t)) \in (\alpha_j - \Delta \alpha_j, \hat{\beta}_j + \Delta \beta_j) $ and $ y_i(t^*, u_i(t)) \in (\alpha_i - \Delta \alpha_i, \hat{\beta}_i + \Delta \beta_i) $. That is, if there is no feasible schedule for Problem 5.1, there always exists $ t^* $ such that $ y(t^*) \in \hat{B} $. This completes the proof. $ \square $

**Proposition A.3.** If a feasible schedule $ \mathcal{T} $ for Problem 5.1 exists, then an input $ u(t) $ for Problem 3.1 always exists.

**Proof.** For any $ i \neq j $, if we have $ T_i \leq T_j $ for $ T_i \in [R_i, D_i] $ and $ T_j \in [R_j, D_j] $, then $ \mathcal{P}_i(T_i) \leq T_j $. From Proposition A.1, there are input signals $ u_i(t) $ and $ u_j(t) $ as defined in the proof of Proposition A.2.

At $ t^* = \mathcal{P}_i(T_i)\Delta t $, vehicle $ i $ exits the intersection by definition of $ \mathcal{P}_i(T_i) $, i.e., $ y_i(t^*, u_i(t)) \geq \beta_i $. At the same time, because $ t^* = \mathcal{P}_i(T_i) \leq T_j $, vehicle $ j $ still does not enter the intersection, i.e., $ y_j(t^*, u_j(t)) \leq y_j(T_j\Delta t, u_j) \leq \alpha_j $. This implies that $ u_i(t) $ and $ u_j(t) $ have the vehicles avoid the bad set $ B $ in (3.1). $ \square $
Appendix B

Total modularity of the Schedule Optimization Problem

Recall the LP relaxation of the Schedule Optimization Problem:

**Problem 5.6 [LP relaxation of Problem 5.5]** Find a vector $\mathcal{T}^\pm = (\mathcal{T}^+, \mathcal{T}^-)'$ such that

\[
\min_{\mathcal{T}^+, \mathcal{T}^-} \sum_{i=1}^{n} (\mathcal{T}^+_i + \mathcal{T}^-_i)
\]

subject to

\[
\mathcal{R}_i \leq (\mathcal{T}^+_i - \mathcal{T}^-_i) + \mathcal{T}_d,i \leq \mathcal{D}_i, \quad \forall i
\]

\[
(\mathcal{T}^+_i - \mathcal{T}^-_i) + \mathcal{T}_d,\pi_i + \vartheta_{\text{max}} \leq (\mathcal{T}^+_j - \mathcal{T}^-_j) + \mathcal{T}_d,\pi_j, \quad \forall i < j,
\]

\[
\mathcal{T}^+_i \geq 0, \mathcal{T}^-_i \geq 0, \quad \forall i.
\]

We rearrange Problem 5.6 to write it as $\min\{c'\mathcal{T}^\pm : \mathcal{A}\mathcal{T}^\pm \leq \mathbf{b}\}$. Since the objective function is the sum of $\mathcal{T}^+_i$ and $\mathcal{T}^-_i$ for all $i$, we have $c = (1, 1, \ldots, 1)$. The
constraints can be written as follows:

\[-T_i^+ + T_i^- \leq -\mathcal{R}_i + \mathcal{T}_{d,i}, \forall i, \quad \text{(B.1)}\]
\[
T_i^+ - T_i^- \leq \mathcal{D}_i - \mathcal{T}_{d,i}, \forall i, \quad \text{(B.2)}
\]
\[
(T_{\pi_i^+} - T_{\pi_i^-}) - (T_{\pi_j^+} - T_{\pi_j^-}) \leq -\theta_{\text{max}} - \mathcal{T}_{d,\pi_i} + \mathcal{T}_{d,\pi_j}, \forall i < j, \quad \text{(B.3)}
\]
\[
-T_i^+ \leq 0, -T_i^- \leq 0, \forall i. \quad \text{(B.4)}
\]

Constraint (B.3) contains \((n - 1)\) inequalities with \(\{(\pi_1, \pi_2), (\pi_2, \pi_3), \ldots, (\pi_{n-1}, \pi_n)\}\).

To generalize Problem 5.6 by considering preceding vehicles that are \(y_i[0] \geq \alpha_i - \Delta \alpha_i\), we define \(\mathcal{P}_0\) as the maximum process time \(\mathcal{P}(\mathcal{T}_i)\) among the vehicles satisfying \(y_i[0] \geq \alpha_i - \Delta \alpha_i\). Thus, we consider an additional inequality \(-(T_{\pi_i^+} - T_{\pi_i^-}) \leq -\mathcal{P}_0 + \mathcal{T}_{d,\pi_i}\).

For the aggregate vectors \(\mathcal{T}^+\) and \(\mathcal{T}^-\) and for the column vector \(\mathcal{T}^\pm := (\mathcal{T}^+, \mathcal{T}^-)'\), these constraints (B.1)-(B.4) can be notated by \(\mathcal{A}_i \mathcal{T} \leq \mathcal{b}_i\) for \(i = \{1, 2, 3, 4\}\), respectively. Notice that

\[
\mathcal{A}_1 = [-\mathcal{I}_{nxn}, \mathcal{I}_{nxn}], \quad \mathcal{A}_2 = [\mathcal{I}_{nxn}, -\mathcal{I}_{nxn}],
\]
\[
\mathcal{A}_3 = [\mathcal{L}, -\mathcal{L}], \quad \mathcal{A}_4 = \mathcal{I}_{2nx2n},
\]

where

\[
L := \begin{bmatrix}
-1 & 0 & 0 & \cdots & 0 \\
1 & -1 & 0 & \cdots & 0 \\
0 & 1 & -1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 1 & -1
\end{bmatrix} \quad \text{(B.5)}
\]

Let \(\mathcal{b}_{i,j}\) denote the \(j\)th element of a vector \(\mathcal{b}_i\) for \(i \in \{1, 2, 3, 4\}\). Then, for \(j \in \{1, \ldots, n\}\),

\[
\mathcal{b}_{1,j} = -\mathcal{R}_{\pi_j} + \mathcal{T}_{d,\pi_j}, \quad \quad \mathcal{b}_{2,j} = \mathcal{D}_{\pi_j} - \mathcal{T}_{d,\pi_j},
\]
and

\[ b_{3,1} = -p_0 + T_{d,n}, \]
\[ b_{3,j} = -\theta_{\max} - T_{d,nj-1} + T_{d,nj}, \text{ for } j \in \{2, \ldots, n\}, \]
\[ b_{4,j} = 0, \text{ for } j \in \{1, \ldots, 2n\}. \]

Notice that for all \( i \in \{1, 2, 3, 4\} \), \( A_i \) are integral matrices with elements of 1, 0, or -1, and \( b_i \) are integral vectors. To prove that the aggregate matrix \( A = [A_1; A_2; A_3; A_4] \) is totally modular, we employ the following proposition from [20].

**Proposition B.1.** Let \( A \) be a totally unimodular matrix. Then the following matrices are all totally unimodular:

\[ -A, A^T, [A, I], [A, -A]. \]

By virtue of Proposition B.1, we can confirm that if \( L \) is totally unimodular, then \( A \) is totally unimodular. This is because a totally unimodular matrix \( L \) leads sequentially \([I; L], [-I; I; L], \) and then \([-I, I; I, -I; L, -L]\) to be totally unimodular. Therefore, \( A \) becomes a totally unimodular matrix.

**Lemma B.1.** The square matrix \( L \) is a totally unimodular matrix.

**Proof.** From (B.5), notice that \( L \) has -1 on the diagonal and 1 on the subdiagonal. The other elements are 0. Thus, for any submatrix of \( L \) is always a triangular matrix. Since all elements in \( L \) are either 1, -1, or 0, any subdeterminant of \( L \) is 1, -1, or 0. \( \square \)

Therefore, given the optimization problem \( \min \{c' T^{\pm} : A T^{\pm} \leq b\} \), which is equivalent to Problem 5.6, we confirm that the matrix \( A \) is totally unimodular, and the vectors \( b \) and \( c \) are integral.
Bibliography


