Supervisory Control for Intersection Collision Avoidance in the Presence of Uncontrolled Vehicles*

Heejin Ahn, Alessandro Colombo, and Domitilla Del Vecchio

***Abstract*— This paper describes the design of a supervisory controller (supervisor) that manages controlled vehicles to avoid intersection collisions in the presence of uncontrolled vehicles. Two main problems are addressed: verification of the safety of all vehicles at intersections, and management of the inputs of controlled vehicles. As for the verification, we apply an inserted idle-time scheduling approach [1] where the term "inserted idle-time" represents the time interval when an intersection is deliberately held idle for uncontrolled vehicles to safely cross the intersection. As for the management, the supervisor is least restrictive in the sense that it overrides all controlled vehicles when an safety violation becomes imminent. In a centralized control system, a computational complexity issue arises. We address this with an efficient version of scheduling using an inflated intersection. The solutions assure the safety of uncontrolled as well as controlled vehicles. It also offers the advantages of increasing traffic flow and energy efficiency.**

I. Introduction

In the United States, the most critical and frequent vehicle collisions happens at intersections according to the National Motor Vehicle Crash Causation Survey (NMVCCS) [2]. The research on intersection collision avoidance requires understanding of dynamics of vehicles, information analysis of traffic, and responses of human operators. Since 2010, the U.S. Department of Transportation has been developing the Intelligent Transportation Systems (ITS), which aims to enable wireless connectivity among vehicles, infrastructure, and portable devices [3]. The networked transportation systems stimulate studies on intersection collision avoidance. Hafner et al. studied a safety for two-vehicle systems, one of which is uncontrolled, with imperfect state information [4]. Using scheduling approach with all controlled vehicles, Colombo et al. studied rear-end collisions [5] and Bruni et al. took uncertainties of measurement and process into account [6].

This paper describes the design of a centralized controller to prevent intersection collisions in the presence of multiple uncontrolled vehicles. In centralized control systems, all decisions are made by a single agent to achieve an objective of the systems. The single agent is designed to be least restrictive in the sense that it never intervenes unless constraint violation becomes imminent [7]. We call the single agent a *supervisor*, which is assumed to be able to measure the dynamic states of all vehicles and give commands to controlled vehicles without uncertainties. The controlled vehicles are those equipped with driver assistance systems that enable the supervisor communicate and override the systems, while the others, uncontrolled vehicles, are not so equipped. An inserted idle-time (IIT) scheduling approach [1] is applied to detect upcoming collisions in the presence of uncontrolled vehicles. Depending on the verification of the scheduling, the supervisor intervenes if necessary. In general, computational complexity is a major concern in centralized control systems [8]; the IIT scheduling used in this paper is known to be NP-hard [9]. This problem is addressed based on an efficient version of scheduling using an inflated intersection.

This paper is organized as follows. In Section II, we review the mathematical frameworks of scheduling problem and define the system. Section III presents two main problems, Verification and Supervisory, and Section IV provides the solutions for the problems. To address the complexity of the solutions, an efficient approach is given in Section V. Simulations for 8 vehicles are given in Section VI. In Section VII, the summary of main achievement and some future works are presented.

II. Problem Foundations

A. Mathematical Frameworks

Scheduling problems are described by jobs, machines, and processing characteristics [10]. 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 time r_i, process time p_i, and due date d_i. A schedule is a vector of starting times for all jobs that all constraints in a scheduling problem are satisfied. While most of scheduling literature focuses on non-delay schedules, we consider a more general class of schedules which is an inserted idle-time (IIT) schedule. This schedule is created with considering an IIT, which are an open time interval when a machine is deliberately held idle while at least one job is waiting to be processed [1]. If we say $(\bar{r}_\gamma, \bar{p}_\gamma)$ is an IIT, a necessary condition for an IIT schedule $\mathtt{t}_i$ to be feasible is $(\mathtt{t}_i, \mathtt{t}_i + p_i) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset$.

In this paper, the scheduling problem $1|r_i|L_{max}$, which aims to find a schedule that minimizes lateness, is considered. To prove that the scheduling problem is equivalent to the control problem that we describe in Section III, we consider the decision version of the scheduling problem, which returns a binary answer yes or no. Define the decision problem DEC($1|r_i|L_{max}$,0) as the question "does a given problem

*This work was supported by ...

Heejin Ahn and Domitilla Del Vecchio are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, USA. Email: hjahn@mit.edu and ddv@mit.edu

Alessandro Colombo is with the Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. Email: alessandro.colombo@polimi.it

$1|r_i|L_{max}$ have a schedule to make L_{max} smaller than or equal to 0?". This is stated in a formal manner as follows.

Definition 1 (DEC($1|r_i|L_{max}$,0)): Given a set of n jobs which have a release time r_i, a process time p_i, and a due date d_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 $\mathsf{T} = \{\mathsf{t}_1, \ldots, \mathsf{t}_n\}$ such that for all $i \in \{1, \ldots, n\}$,

$$r_i \leq \mathsf{t}_i \leq d_i - p_i,$$

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

$$\mathsf{t}_i \leq \mathsf{t}_j \Rightarrow \mathsf{t}_i + p_i \leq \mathsf{t}_j,$$

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

$$\mathsf{t}_i \leq \bar{r}_\gamma \Rightarrow \mathsf{t}_i + p_i \leq \bar{r}_\gamma$$
$$\mathsf{t}_i \geq \bar{r}_\gamma \Rightarrow \mathsf{t}_i \geq \bar{p}_\gamma$$

Notice that if such a schedule exists, $(\mathsf{t}_i, \mathsf{t}_i + p_i) \cap (\bar{r}_\gamma, \bar{p}_\gamma) = \emptyset$ because for all $t^* \in (\mathsf{t}_i, \mathsf{t}_i + p_i)$, the last two constraints imply $t^* < \bar{r}_\gamma$ or $t^* > \bar{p}_\gamma$. Also the lateness $L_{max} = t_i + p_i - d_i$ of the schedule becomes non-positive, and DEC($1|r_i|L_{max}$,0) returns *yes*. The problem DEC($1|r_i, p_i = 1|L_{max}$,0) describes the same constraints as Definition 1 except with the addition of constraint $p_i = 1$. It has been shown that a problem with arbitrary process times such as DEC($1|r_i|L_{max}$,0) is NP-hard, while a problem with unit process times such as DEC($1|r_i, p_i = 1|L_{max}$,0) is tractable [11]. The problem DEC($1|r_i, p_i = 1|L_{max}$,0) is used to address a complexity issue in Section V.

To determine the complexity of a different sort of problems and to solve a problem in the presence of a solution of another problem, concepts of "*reduciblity*" and "*equivalence*" [12] 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 [13]. A problem P_1 is reducible to a problem P_2 if an instance I_2 of P_2 can be constructed in polynomial-bounded time for any instance I_1 of P_1 and solving the instance I_2 of P_2 also solves the instance I_1 of P_1. We write $P_1 \propto P_2$ if P_1 is reducible to P_2. If $P_2 \propto P_1$ and $P_1 \propto P_2$, they are equivalent.

B. System Definition

We consider n vehicles moving along unidirectional paths intersecting at one point. We assume that a subset of these vehicles can be controlled while the remaining vehicles cannot be controlled. Consider all vehicles identified with a natural number from 1 to n. To distinguish controlled and uncontrolled vehicles, construct a controlled set $\mathcal{C}$ and an uncontrolled set $\bar{\mathcal{C}}$, which contain n_c and $n_{\bar{c}}$ elements, respectively, as follows.

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

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

In this paper, i usually denotes a controlled vehicle while γ is used for indicating an uncontrolled vehicle.

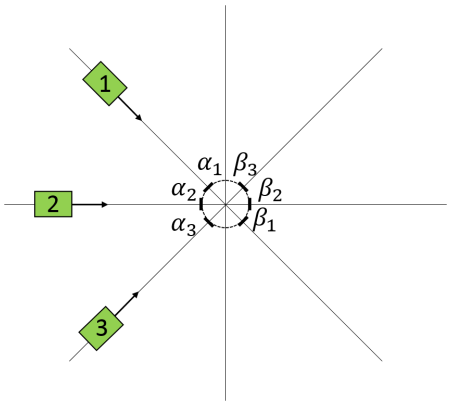

Fig. 1. An example of 3 vehicles approaching an intersection

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

$$\dot{x}_i = f(x_i, u_i) \qquad y_i = h(x_i), \tag{1}$$

while each uncontrolled vehicle by

$$\dot{x}_\gamma = f(x_\gamma, d_\gamma) \qquad y_\gamma = h(x_\gamma). \tag{2}$$

where $x_i \in X_i \subseteq \mathbb{R}^r$, $u_i \in U_i \subset \mathbb{R}^s$, $y_i \in Y_i \subseteq \mathbb{R}$, $x_\gamma \in X_\gamma \subseteq \mathbb{R}^r$, $d_\gamma \in \mathsf{D}_\gamma \subset \mathbb{R}^s$, 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 $\mathcal{U}_i$ and $\mathcal{D}_\gamma$, respectively. The notations $\mathbf{x}, \mathbf{u}, \mathbf{d}, \mathbf{y}$ denote aggregate vectors for these states, inputs, disturbances, and outputs, for instance $\mathbf{x} = \{x_1, \ldots, x_n\}$. The sets containing these vectors are denoted $X, Y, U, \mathsf{D}, \mathcal{U}, \mathcal{D}$. Notice that $X \subseteq (\mathbb{R}^r)^n$ and $Y \subseteq \mathbb{R}^n$ while $U \subset (\mathbb{R}^s)^{n_c}$ and $\mathsf{D} \subset (\mathbb{R}^s)^{n_{\bar{c}}}$. Moreover, the output function $h(x_i)$ and $h(x_\gamma)$ are continuous and the derivative of the output is bounded, $\dot{y}_i \in [\dot{y}_{i,m}, \dot{y}_{i,M}]$ with $\dot{y}_{i,m} > 0$. The ordered sets U_i and D_γ are compact, that is, $u_i \in [u_{i,m}, u_{i,M}]$ and $d_\gamma \in [d_{\gamma,m}, d_{\gamma,M}]$.

The parallel composition of equation (1) and (2) for all n vehicles describes the system dynamics.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \qquad \mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{3}$$

Assume that this system has a unique solution which 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 $x_i(t, u_i, x_i(t_0))$ and $y_i(t, u_i, x_i(t_0))$, respectively. The corresponding aggregate state is denoted by $\mathbf{x}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}(t_0))$ and the output by $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}(t_0))$. For notational brevity, we omit an initial condition when $t_0 = 0$ and omit an input when it is not important. We say maps of ordered sets $\mathcal{U}_i \to X_i$ and $\mathcal{D}_\gamma \to X_\gamma$ are order preserving as defined in [14], that is, if $u_i \leq u_i' \in \mathcal{U}_i$ for $i \in \mathcal{C}$ and $d_\gamma \leq d_\gamma' \in \mathcal{D}_\gamma$ for $\gamma \in \bar{\mathcal{C}}$, then, respectively, $x_i(t, u_i) \leq x_i(t, u_i')$ and $x_\gamma(t, d_\gamma) \leq x_\gamma(t, d_\gamma')$. Also a map from an ordered set of initial states to an ordered set of

outputs, $X_i \to Y_i$ is order preserving: if $x_i(0) \leq x_i'(0)$, then $y_i(t, u_i, x_i(0)) \leq y_i(t, u_i, x_i'(0))$.

III. Problem Statement

In this paper, two main problems are addressed: verification of the safety of all vehicles at intersections, Verification Problem, and management of the inputs of controlled vehicles, Supervisory Problem. The Verification Problem determines whether the current inputs of drivers lead to an intersection collision. If it is true, the Supervisory Problem overrides all controlled vehicles with safe input profiles, which are generated in the previous step.

A. Verification Problem

As seen in figure 1, an interval (α_i, β_i) is assigned to vehicle i for all $i \in \mathcal{C} \cup \bar{\mathcal{C}}$, and let $\alpha := \{\alpha_1, \ldots, \alpha_n\}$ and $\beta := \{\beta_1, \ldots, \beta_n\}$. If two or more vehicles are inside the intersection at the same time, we consider a collision occurs. This subset of the output space Y is called the *bad set* B and defined as follows.

$$B := \{\mathbf{y} \in Y : y_i \in (\alpha_i, \beta_i) \text{ and } y_j \in (\alpha_j, \beta_j) \text{ for some } i \neq j \text{ such that } i \in \mathcal{C} \text{ and } j \in \mathcal{C} \cup \bar{\mathcal{C}} \text{ for any } \mathbf{d}\}.$$

Throughout this paper, we focus on two kinds of collisions: between controlled vehicles and between controlled and uncontrolled ones. Now we can formalize the Verification Problem.

Problem 1 (Verification Problem): Given an initial condition $\mathbf{x}(0)$, determine if there exists an input signal $\mathbf{u}(t) \in \mathcal{U}$ which guarantees that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \notin B$ for all $\mathbf{d}(t) \in \mathcal{D}$ for all $t \geq 0$.

Input $\mathbf{u}(t) \in \mathcal{C}$ ensures safety if and only if $\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \notin B$ "for all" $\mathbf{d}(t) \in \mathcal{D}$ for all $t \geq 0$. An instance of Problem 1 is described by the initial condition $\mathbf{x}(0)$ and the parameters $\Theta = \{f, h, X, Y, U, \mathsf{D}, \mathcal{U}, \mathcal{D}, \alpha, \beta\}$. Thus, if there exists $\mathbf{u}(t)$ which ensures safety for a given instance $\{\mathbf{x}(0), \Theta\}$, then we say $\{\mathbf{x}(0), \Theta\} \in$ Problem 1.

To solve Problem 1, we adapt the IIT scheduling problem DEC$(1|r_i|L_{max}, 0)$ in Definition 1. We prove the equivalence of both problems as defined in Section II: the intersection plays the role of the machine, and the n_c vehicles are the jobs. Processing characteristics are defined as follows.

Definition 2: For all $i \in \mathcal{C}$ where $y_i(0) < \alpha_i$, let

$$R_i := \min_{u_i \in \mathcal{U}_i} \{t \geq 0 : y_i(t, u_i) = \alpha_i\}$$

$$D_i := \max_{u_i \in \mathcal{U}_i} \{t \geq 0 : y_i(t, u_i) = \alpha_i\}.$$

Given a real number T_i, let

$$P_i(T_i) := \min_{u_i \in \mathcal{U}_i} \{t \geq 0 : y_i(t, u_i) = \beta_i \text{ with a constraint } y_i(T_i, u_i) = \alpha_i\}.$$

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

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

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

We call $\bar{R}_\gamma$ and $\bar{P}_\gamma$ the idle-time variables. 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 $R_i = 0, D_i = 0$, and $P_i(T_i) = \min_{u_i \in \mathcal{U}_i}\{t : y_i(t, u_i) = \beta_i\}$. If the constraint is not satisfied, set $P_i(T_i) = \infty$. For $\gamma \in \bar{\mathcal{C}}$, 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 $\bar{P}_\gamma = \max_{d_\gamma \in \mathcal{D}_\gamma}\{t : y_\gamma(t, d_\gamma) = \beta_\gamma\}$.

These variables are well-defined since it is assumed that the system (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 $\gamma \in \bar{\mathcal{C}}$ are fixed once an initial condition is provided. We consider $(\bar{R}_\gamma, \bar{P}_\gamma)$ for $\gamma \in \bar{\mathcal{C}}$ to be an IIT because the intersection is kept empty as long as an uncontrolled vehicle stays inside the intersection for all d_γ. A schedule T_i and a process time $P_i(T_i)$ are chosen to satisfy following conditions.

Problem 2 (IIT Scheduling Problem): Given an initial condition $\mathbf{x}(0)$, determine whether there exists a schedule $\mathbf{T} = \{T_i : i \in \mathcal{C}\} \in \mathbb{R}_+^{n_c}$ such that for all $i \in \mathcal{C}$,

$$R_i \leq T_i \leq D_i \tag{4}$$

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

$$T_i \leq T_j \Rightarrow P_i(T_i) \leq T_j \tag{5}$$

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

$$T_i \leq \bar{R}_\gamma \Rightarrow P_i(T_i) \leq \bar{R}_\gamma \tag{6}$$

$$T_i \geq \bar{R}_\gamma \Rightarrow T_i \geq \bar{P}_\gamma \tag{7}$$

This problem is also described by an instance $I = \{\mathbf{x}(0), \Theta\}$. A feasible schedule for Problem 2 ensures $(T_i, P_i(T_i)) \cap (\bar{R}_\gamma, \bar{P}_\gamma) = 0$ by (6) and (7). The IIT scheduling determines the safety of all vehicles by sequencing controlled vehicles to cross an intersection before or after IIT. We prove that Problem 2 is equivalent to Problem 1.

Theorem 1: Problem 1 and Problem 2 are equivalent.

Proof: We prove that an instance $I = \{\mathbf{x}(0), \Theta\}$ belongs to the Verification Problem (Problem 1) if and only if it belongs to the IIT Scheduling Problem (Problem 2).

$$I \in \text{Verification Problem} \Leftrightarrow I \in \text{Scheduling Problem}$$

($\Rightarrow$) Given an initial condition $\mathbf{x}(0)$, there is an input $\tilde{\mathbf{u}}$ which ensures safety, that is, $\mathbf{y}(t, \tilde{\mathbf{u}}, \mathbf{d}) \notin B$ for all $\mathbf{d}$ for all $t \geq 0$.

For $y_i(0) < \alpha_i$ for $i \in \mathcal{C}$, define T_i as the time t when $y_i(t, \tilde{u}_i) = \alpha_i$. Also define $\tilde{P}_i(T_i)$ as the time when $y_i(t, \tilde{u}_i) = \beta_i$ with constraints that $y_i(T_i, \tilde{u}_i) = \alpha_i$. If $y_i(0) \geq \alpha_i$, set $T_i = 0$. Set $\tilde{P}_i(T_i) = 0$ if $y_i(0) \geq \beta_i$. By the definitions of R_i and D_i, $R_i \leq T_i \leq D_i$ (Condition (4)). Suppose $T_i \leq T_j$ for some $i \neq j \in \mathcal{C}$. Since $\tilde{u}_i$ and $\tilde{u}_j$ prevent an intersection collision between vehicles i and j, when $y_j(t, \tilde{u}_j) = \alpha_j$, $y_i(t, \tilde{u}_i) \geq \beta_i$. Thus, $\tilde{P}_i(T_i) \leq T_j$. By the definition of $P_i(T_i)$, $P_i(T_i) \leq \tilde{P}_i(T_i) \leq T_j$ (Condition (5)). Also, with the assumption that the input $\tilde{\mathbf{u}}$ ensures safety for all $\mathbf{d}$, if $T_i \leq \bar{R}_\gamma$ for $i \in \mathcal{C}, \gamma \in \bar{\mathcal{C}}$, 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 $\bar{R}_\gamma$ by definition. Combining this with the definition of $P_i(T_i)$,

$P_i(T_i) \leq \tilde{P}_i(T_i) \leq \bar{R}_\gamma$ (Condition (6)). If $T_i \geq \bar{R}_\gamma$, when $y_i(t, \tilde{u}_i) = \alpha_i$, the vehicle γ must stay away from the intersection, i.e., $\min_{d_\gamma} y_\gamma(t, d_\gamma) \geq \beta_\gamma$. Because of the increasing property of the output with respect to t, $T_i \geq \bar{P}_\gamma$ (Condition (7)).

($\Leftarrow$) Given the same initial condition $\mathbf{x}(0)$, there exists a schedule $\mathbf{T} \in \mathbb{R}^{n_c}$ that satisfies all conditions of Problem 2.

We start assuming that $y_i(0) < \alpha_i$ for all $i \in \mathcal{C} \cup \bar{\mathcal{C}}$, which does not break generality since vehicles after the intersection are not of interest. According to Lemma 5.1 in [15], with the assumptions that $\mathcal{U}_i$ is path connected for $i \in \mathcal{C}$, that the system (3) has a unique solution dependent continuously on an input, and that the output is continuous, if $T_i \in [R_i, D_i]$ for $y_i < \alpha_i$, there exists an input $u_i \in \mathcal{U}_i$ such that $y_i(T_i, u_i) = \alpha_i$. If these assumptions hold, we are able to construct $u_i \in \mathcal{U}_i$ such that $y_i(T_i, u_i) = \alpha_i$ for all $i \in \mathcal{C}$.

Now, we show this input $u_i \in \mathbf{u}$ such that $\mathbf{u}$ ensures safety, if T_i satisfies condition (4)-(7). When $T_i \leq T_j$ for $i \neq j \in \mathcal{C}$, condition (5) indicates $P_i(T_i) \leq T_j$, which implies 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. In condition (6), when $T_i \leq \bar{R}_\gamma$ for $i \in \mathcal{C}, \gamma \in \bar{\mathcal{C}}$, $P_i(T_i) \leq \bar{R}_\gamma$. This implies that when $\max_{d_\gamma} y_\gamma(t, d_\gamma) = \alpha_\gamma$, we have $y_i(t, u_i) \geq \beta_i$: in other words, $y_\gamma(t, d_\gamma) \leq \alpha_\gamma$ and $y_i(t, u_i) \geq \beta_i$. If $T_i \geq \bar{R}_\gamma$, the condition (7) indicates 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_γ that $y_\gamma(t, d_\gamma) \geq \beta_\gamma$. From these conditions (6) and (7), it is shown that u_i prevents collision between controlled and uncontrolled vehicles for all d_γ. ■

B. Supervisory Problem

We now design a supervisor operating in discrete time. At each step $k\tau$, the current state $\mathbf{x}(k\tau) \in X$ and the desired input $\mathbf{a}_k \in U$ are given, where a desired input represents the input applied by a driver. Then, the supervisor returns $\mathbf{u}_{k,out}$ depending on the state prediction for the next step, which is defined as follows.

$$\mathbf{x}_{k+1}(\mathbf{a}_k) := \begin{cases} x_i(\tau, a_{i,k}, x_i(k\tau)) & i \in \mathcal{C} \\ x_\gamma(k\tau) & \gamma \in \bar{\mathcal{C}} \end{cases} \tag{8}$$

where $\mathbf{a}_k = \{a_{i,k} \in U_i : i \in \mathcal{C}\}$. Notice that the current disturbance $\mathbf{d}_k \in \mathcal{D}$ does not contribute to the state prediction since uncontrolled vehicles are not equipped with driver assistance systems that can monitor the input of the drivers.

Because a supervisor interacts with human operators, it has to be least restrictive in the sense that it intervenes if any future collision is detected. To describe future events, define a continuous input profile $\mathbf{u}_k^\infty(t)$ defined on $t \in [k\tau, \infty)$; let $\mathbf{u}_k(t)$ be $\mathbf{u}_k^\infty(t)$ restricted to $t \in [k\tau, (k+1)\tau]$. During this time period, let $\mathbf{u}_k(t) = \mathbf{a}_k$, where $\mathbf{a}_k$ is a desired input by drivers at $t = k\tau$ and is constant on $[k\tau, (k+1)\tau]$. Similarly, let $\mathbf{u}_{k,safe}^\infty(t)$ be a safe input defined on $t \in [k\tau, \infty)$; let $\mathbf{u}_{k,safe}(t)$ be $\mathbf{u}_{k,safe}^\infty(t)$ restricted to $t \in [k\tau, (k+1)\tau]$. We now formalize the Supervisory Problem.

Problem 3 (Supervisory Problem): Design a supervisor $s(\mathbf{x}(k\tau), \mathbf{a}_k)$ at time step $k\tau$ such that

$$s(\mathbf{x}(k\tau), \mathbf{a}_k) = \begin{cases} \mathbf{u}_k & \text{if } \exists \mathbf{u}_k^\infty : \mathbf{y}(t, \mathbf{u}_k^\infty, \mathbf{d}) \notin B \\ & \text{for all } \mathbf{d} \in \mathcal{D} \text{ for } t \geq k\tau \\ \mathbf{u}_{k,safe} & \text{otherwise} \end{cases}$$

and such that it is non-blocking: if $s(\mathbf{x}(k\tau), \mathbf{a}_k) \neq \emptyset$, then for any $\mathbf{a}_{k+1}, s(\mathbf{x}((k+1)\tau), \mathbf{a}_{k+1}) \neq \emptyset$.
The non-blocking property assures that the supervisor is least restrictive because it enables the supervisor accept the inputs of drivers immediately after intervention on $[k\tau, (k+1)\tau]$.

IV. Problem Solutions

A. Solution of Problem 1

Assume, without loss of generality, that for m controlled vehicles, $y_i \geq \alpha_i$. Let $\mathcal{P}$ be all permutations of $i \in \mathcal{C}$ such that $y_i < \alpha_i$, and then each element of $\mathcal{P}$ becomes $(n_c - m)$-tuple, which is denoted by π. Set $\bar{\pi}$ a $n_{\bar{c}}$-tuple composed of all $\gamma \in \bar{\mathcal{C}}$ in an increasing order of $\bar{R}_\gamma$. The notation π_i denotes i-th index of π and $\bar{\pi}_i$ denotes i-th index of $\bar{\pi}$. For the moment, set $\delta = 0$; its usage and parametric values become clear in Section IV-B.

Algorithm 1 Verification of the safety of all vehicles

procedure ExactSolution($\mathbf{x}(0), \delta$)
  if $\mathbf{y}(0) \in B$ **then return** $\{\mathbf{0}, no\}$
  for all $i \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$ **do**
    given $x_i(0)$ calculate R_i, D_i
    given $x_\gamma(0)$ calculate $\bar{R}_\gamma, \bar{P}_\gamma$
    $\bar{R}_\gamma \leftarrow \max(\bar{R}_\gamma - \delta, 0)$
    $\bar{P}_\gamma \leftarrow \max(\bar{P}_\gamma - \delta, 0)$
  for $i \in \mathcal{C}$ such that $y_i(0) \geq \alpha_i$ **do**
    $T_i \leftarrow 0$
    calculate $P_i(T_i)$ and $P_{max} \leftarrow \max_i P_i(T_i)$
  for all $\pi \in \mathcal{P}$ **do**
    for $j \leftarrow 1$ to $n_c - m$ **do**
      $T_{\pi_1} \leftarrow \max(R_{\pi_1}, P_{max})$ for $j = 1$
      $T_{\pi_j} \leftarrow \max(R_{\pi_j}, P_{\pi_{j-1}}(T_{\pi_{j-1}}))$
      given T_{π_j} calculate P_{π_j}
      for $i \leftarrow 1$ to $n_{\bar{c}}$ **do**
        if $T_{\pi_j} \geq \bar{R}_{\bar{\pi}_i}$ **then**
          $T_{\pi_j} \leftarrow \max(T_{\pi_j}, \bar{P}_{\bar{\pi}_i})$
          given T_{π_j} calculate P_{π_j}
        else if $P_{\pi_j} > \bar{R}_{\bar{\pi}_i}$ **then**
          $T_{\pi_j} \leftarrow \bar{P}_{\bar{\pi}_i}$
          given T_{π_j} calculate P_{π_j}
  if $T_i \leq D_i$ for all $i \in \mathcal{C}$ **then**
    return $\{\mathbf{T}, yes\}$

Notice that given an initial condition $\mathbf{x}(0)$ with $\delta = 0$, Algorithm 1 becomes the solution for Problem 2 and for Problem 1 by Theorem 1. Also notice that computational complexity increases in a factorial manner since the algorithm operates all permutations in $\mathcal{P}$ to return *no*. Therefore,

the algorithm is intractable if n is large; in other words, the solution for Problem 1 is NP hard.

Example 1: Suppose we have $\mathcal{C} = \{1,3,4\}$ and $\bar{\mathcal{C}} = \{2,5\}$ that are modelled for simplicity as $f(x_i,u_i) = u_i, h(x_i) = x_i$ for $i \in \mathcal{C}$, and $f(x_\gamma,d_\gamma) = d_\gamma, h(x_\gamma) = x_\gamma$ for $\gamma \in \bar{\mathcal{C}}$. A given initial condition is $\mathbf{x}(0) = \{44,26,20,5,2\}$ 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 processing characteristics become $[R_1,R_3,R_4] = [\frac{2}{5},2,3]$, $[D_1,D_3,D_4] = [2,10,15]$, $[\bar{R}_2,\bar{R}_5] = [2,4]$, 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 $P_1 = T_1 + \frac{\beta_1-\alpha_1}{u_M} = \frac{3}{5}$. Since $T_1 < \bar{R}_2 < \bar{R}_5$ and $P_1 < \bar{R}_2 < \bar{R}_5$, it is determined that $T_1 = \frac{2}{5}$ and $P_1 = \frac{2}{5}$. 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 $i = 1$ thus $\bar{\pi}_1 = 2$, we have $T_3 \geq \bar{R}_2$ and thus $T_3 = \max(T_3,\bar{P}_2) = 4\frac{1}{2}$. This schedule leads to $P_3 = 4\frac{7}{10}$. For $i = 2$ then $\bar{\pi}_2 = 5$, although $T_3 < \bar{R}_5$, we have $P_3 > \bar{R}_5$ so that $T_3 = \bar{P}_5 = 8\frac{1}{2}$ and $P_3 = 8\frac{7}{10}$. Finally for $j = 3$ thus $\pi_3 = 4$, we have $T_4 = \max(R_4,P_3) = 8\frac{7}{10}$ and $P_4 = 8\frac{9}{10}$. Even though for $i = 1$ we have $T_4 \geq \bar{R}_2$, notice that T_4 does not change because $\max(T_4,\bar{R}_2) = T_4$. This is same for $i = 2$. Since $T_i \leq D_i$ for all $i \in \{1,3,4\}$, we have found a set of schedule $\mathbf{T} = \{\frac{2}{5},8\frac{1}{2},8\frac{7}{10}\}$. If there was no feasible schedule for this π, Algorithm 1 would try the other permutations such as $\pi = [1,4,3]$.

B. Solution of Problem 3

A supervisor is designed to override controlled vehicles with a safe input if a desired input, which represents a driver's input, 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 $i \in \mathcal{C}$ as follows.

$$\sigma_i(x_i(0),T_i) := \arg \inf_{u_i \in \mathcal{U}_i} \{t \geq 0 : y_i(t,u_i) = \beta_i \text{ for } i \in \mathcal{C} \text{ with constraint } y_i(T_i,u_i) = \alpha_i.\} \quad (9)$$

This operator returns an input $u_i \in \mathcal{U}_i$ for $i \in \mathcal{C}$ such that $y_i(t,u_i)$ reaches α_i at T_i and β_i at $P_i(T_i)$. Set $\sigma_i(x_i(0),T_i) = u_{i,M}$ if $y_i(0) \geq \alpha_i$. If $y_i(0) \geq \beta_i$ or if the constraint cannot be satisfied, set $\sigma_i(x_i(0),T_i) = \emptyset$. Let $\sigma(\mathbf{x}(\mathbf{0}),\mathbf{T})$ be a parallel composition of Equation (9) for all $i \in \mathcal{C}$.

At each time step, Problem 1 determines whether a predicted state from a given input leads to an intersection collision using Algorithm 1. However, in the state prediction $\mathbf{x}_{k+1}(\mathbf{a}_k)$, we do not predict the states for uncontrolled vehicles because $\mathbf{d}_k$ is not measured. Thus, a time delay δ occurs for vehicle γ for $\gamma \in \bar{\mathcal{C}}$. Let $(\bar{R}_\gamma^k,\bar{P}_\gamma^k)$ denote an IIT for a given initial condition $x_\gamma(k\tau)$. Then a predicted idle-time $(\bar{R}_\gamma^{k+1},\bar{P}_\gamma^{k+1})$ has to preserve the IIT except the passed time step τ. Thus,

$$(\bar{R}_\gamma^{k+1},\bar{P}_\gamma^{k+1}) = (\bar{R}_\gamma^k - \tau, \bar{P}_\gamma^k - \tau). \quad (10)$$

Notice that the time delay δ is the parameters of either 0 or τ. When Algorithm 1 accepts the state prediction, $\delta = \tau$; otherwise, set $\delta = 0$.

Algorithm 2 Solution of Problem 3

procedure SUPERVISOR($\mathbf{x}(k\tau),\mathbf{a}_k$)
  $\mathbf{u}_k(t) \leftarrow \mathbf{a}_k \ \forall t \in [k\tau,(k+1)\tau]$
  $\{\mathbf{T},answer\} \leftarrow$ EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_k),\tau$)
  if $answer = yes$ **then**
    $\mathbf{u}_{k+1,safe}^{\infty} \leftarrow \sigma(\mathbf{x}_{k+1}(\mathbf{u}_k),\mathbf{T})$
    $\mathbf{u}_{k+1,safe} \leftarrow \mathbf{u}_{k+1,safe}^{\infty}$restricted to$[k\tau,(k+1)\tau]$
    return $\mathbf{u}_k$
  else
    $\{\mathbf{T},answer\}$
      $\leftarrow$ EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_{k,safe}),\tau$)
    $\mathbf{u}_{k+1,safe}^{\infty} \leftarrow \sigma(\mathbf{x}_{k+1}(\mathbf{u}_{k,safe}),\mathbf{T})$
    $\mathbf{u}_{k+1,safe} \leftarrow \mathbf{u}_{k+1,safe}^{\infty}$restricted to$[k\tau,(k+1)\tau]$
    return $\mathbf{u}_{k,safe}$

Lemma 1: If EXACTSOLUTION($\mathbf{x}(k\tau),0$) $= \{\mathbf{T},yes\}$, then $\mathbf{u} := \sigma(\mathbf{x}(k\tau),\mathbf{T})$ exists. Moreover, the state prediction (8) from $\mathbf{u}_k$ which restricts $\mathbf{u}$ to $[k\tau,(k+1)\tau]$ leads to EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_k),\tau$) $= \{\mathbf{T},yes\}$. Moreover, $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k),\mathbf{T}) \neq \emptyset$.

Proof: Given an initial condition $\mathbf{x}(k\tau)$, if a schedule $\mathbf{T}$ exists, then $\{\mathbf{x}(k\tau),\Theta\} \in$ Problem 1 by Theorem 1, indicating that there exists an input $\mathbf{u} \in \mathcal{U}$ ensuring safety on $t \in [k\tau,\infty)$ for all $\mathbf{d} \in \mathcal{D}$. Thus, $\sigma(\mathbf{x}(k\tau),\mathbf{T}) \neq \emptyset$. Notice that $\mathbf{y}(t,\mathbf{u},\mathbf{d},\mathbf{x}(k\tau)) \notin B$ is equivalent to the conditions that $T_i \in [R_i,D_i]$, that $(T_i,P_i(T_i))$ does not intersect each other for all $i \in \mathcal{C}$, and that $(T_i,P_i(T_i)) \cap (\bar{R}_\gamma^k,\bar{P}_\gamma^k) = \emptyset$ for all $\gamma \in \bar{\mathcal{C}}$ by Theorem 1. Let $\tilde{\mathbf{u}}$ be $\mathbf{u}$ restricted to $[(k+1)\tau,\infty)$. If we say $\mathbf{x}_{k+1}(\mathbf{u}_k)$ has a schedule T_i' for all $i \in \mathcal{C}$, then $T_i' = T_i - \tau$ and $P_i'(T_i') = P_i(T_i) - \tau$ since $\mathbf{u} = \mathbf{u}_k \cup \tilde{\mathbf{u}}$. Consequently, intervals $(T_i',P_i'(T_i'))$ never overlap for all $i \in \mathcal{C}$ and $(T_i',P_i'(T_i')) \cap (\bar{R}_\gamma^{k+1},\bar{P}_\gamma^{k+1}) = \emptyset$ for all $\gamma \in \bar{\mathcal{C}}$ due to (10). Therefore, EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_k),\tau$) returns *yes*. ■

Lemma 2: If EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{a}_k),\tau$) $= \{\mathbf{T},yes\}$, then $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k),\mathbf{T}) \neq \emptyset$.

Proof: Define $\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k)$ as $x_\gamma(t,d_{\gamma,k},x_\gamma(k\tau))$ for any $d_\gamma \in \mathcal{D}_\gamma$ for all $\gamma \in \bar{\mathcal{C}}$, and as the same as $\mathbf{x}_{k+1}(\mathbf{a}_k)$ for all $i \in \mathcal{C}$. Let $(\tilde{R}_\gamma^{k+1},\tilde{P}_\gamma^{k+1})$ denote the corresponding idle-time to $x_\gamma(t,d_{\gamma,k},x_\gamma(k\tau))$. Notice that $(\tilde{R}_\gamma^{k+1},\tilde{P}_\gamma^{k+1}) \subseteq (\bar{R}_\gamma^{k+1},\bar{P}_\gamma^{k+1})$ because the latter considers all possible idle-times for all d_γ as can be seen from (10). If EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{a}_k),\tau$) returns *yes*, a schedule $\mathbf{T}$ satisfies that $T_i \in [R_i,D_i]$, that $(T_i,P_i(T_i))$ does not intersect each other for all $i \in \mathcal{C}$, and that $(T_i,P_i(T_i)) \cap (\bar{R}_\gamma^{k+1},\bar{P}_\gamma^{k+1}) = \emptyset$ for all $\gamma \in \bar{\mathcal{C}}$. From the last condition, we also have $(T_i,P_i(T_i)) \cap (\tilde{R}_\gamma^{k+1},\tilde{P}_\gamma^{k+1}) = \emptyset$. Therefore, this schedule $\mathbf{T}$ makes EXACTSOLUTION($\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k),0$) return *yes*. From the first part of Lemma 1, $\sigma(\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k),\mathbf{T}) \neq \emptyset$. Since $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k),\mathbf{T})$ is composed of σ_i for all $i \in \mathcal{C}$, and $\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k)$ is equivalent to $\mathbf{x}_{k+1}(\mathbf{a}_k)$ for all $i \in \mathcal{C}$, we prove that $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k),\mathbf{T}) \neq \emptyset$. ■

Theorem 2: The supervisor $s(\mathbf{x}(k\tau),\mathbf{a}_k)$ defined in Algorithm 2 solves Problem 3.

Proof: If EXACTSOLUTION($\mathbf{x}_{k+1}(\mathbf{a}_k),\tau$) returns *yes*,

by Lemma 2 there exists $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k), \mathbf{T})$ that ensures safety; thus, Algorithm 2 returns $\mathbf{u}_k$ which is identical to $\mathbf{a}_k$. If EXACTSOLUTION$(\mathbf{x}_{k+1}(\mathbf{a}_k), \tau)$ returns *no*, Algorithm 2 returns $\mathbf{u}_{k,safe}$. Thus, the supervisor in the Algorithm 2 is designed to solve the Problem 3.

The key proof is to show its non-blocking property. We use mathematical induction to prove this. Assume that $\mathbf{u}_{0,out} \neq \emptyset$. Suppose at $t = k\tau$, $s(\mathbf{x}(k\tau), \mathbf{a}_k) = \mathbf{u}_{k,out}$ and $\mathbf{u}_{k,out} \neq \emptyset$. Then we must prove that $s(\mathbf{x}_{k+1}(\mathbf{u}_{k,out}), \mathbf{a}_{k+1}) \neq \emptyset$ no matter what $\mathbf{a}_{k+1}$ is applied. Notice that this is possible when $\mathbf{u}^{\infty}_{k+1,safe} = \sigma(\mathbf{x}_{k+1}(\mathbf{u}_{k,out}), \mathbf{T}))$ exists. As seen in the Algorithm 2, $\mathbf{u}_{k,out}$ is either $\mathbf{u}_k$ or $\mathbf{u}_{k,safe}$. In the former case, EXACTSOLUTION$(\mathbf{x}_{k+1}(\mathbf{u}_k), \tau) = \{\mathbf{T}, yes\}$, and by Lemma 2, $\sigma(\mathbf{x}_{k+1}(\mathbf{u}_k), \mathbf{T})$ is non-empty. In the latter case, because we assume by induction that $\mathbf{u}^{\infty}_{k,safe}$ exists, by Lemma 1, EXACTSOLUTION$(\mathbf{x}_{k+1}(\mathbf{u}_{k,safe}), \tau)$ returns *yes*. In turn, Lemma 2 says $\mathbf{u}^{\infty}_{k+1,safe}$ is defined. Therefore, the supervisor is non-blocking. ■

Throughout this paper, we assume that EXACTSOLUTION$(\mathbf{x}(0), 0) = \{\mathbf{T}, yes\}$. If this assumption does not hold, the whole processes in Algorithm 2 does not work.

V. EFFICIENT APPROACH

As seen from Algorithm 1, the control problem in Section III may not be solved with large n. Therefore, we propose an efficient but approximate version of Problem 2, and design the corresponding supervisor.

A. Efficient Scheduling Problem and Solution

Garey et al. [9] proposed an efficient algorithm to solve DEC$(1|r_i, p_i = 1|L_{max}$,0), in which all jobs have a unit process time, and other constraints are the same as Definition 1. To design the algorithm, they introduce the concept of a forbidden region $\mathbf{F}$, a time interval when no task is allowed to start to have a feasible schedule. This is reported in Algorithm 3. The difference between Algorithm 3 and Algorithm A in [9] is that the latter initially declares $\mathbf{F}$ an empty set at the beginning. Let $\mathbf{F}_k \subset \mathbb{R}$ indicate the kth interval in $\mathbf{F}$.

We give a simple example to understand this algorithm.

Example 2: Consider 3 jobs on a machine with $\mathbf{r} = [7, 7, 8]$, $\mathbf{d} = [12, 12, 10]$, and unit process time. An initially declared forbidden region is $\mathbf{F} = \{(8\frac{1}{2}, 10\frac{1}{2})\}$. The increasing order of $\mathbf{r}$ is [1,2,3], so the Algorithm 3 starts from $i = 3$. For $d_1, d_2, d_3 \geq d_3$, since $d_1 - 1 = d_2 - 1 \notin \mathbf{F}$, we have $c_1 = 11, c_2 = 11$ while $c_3 = \inf(\mathbf{F}_1) = 8\frac{1}{2}$ because $d_3 - 1 \in \mathbf{F}_1$. Since $r_2 < r_3$, we find $c = 8\frac{1}{2}$ which is inside $[r_3, r_3 + 1)$, and thus $\mathbf{F} = \{(7\frac{1}{2}, 8), (8\frac{1}{2}, 10\frac{1}{2})\}$. 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 $c_1 = c_2 = 7\frac{1}{2}$, it is determined that $\mathbf{F} = \{(6\frac{1}{2}, 7), (7\frac{1}{2}, 8), (8\frac{1}{2}, 10\frac{1}{2})\}$. Then the next step allocates a schedule to each job. For $i = 1$, $s = 7$ and $\mathsf{t}_1 = 7$. For $i = 2$, we have that $j = 3$ since d_3 is the least due date and $r_3 \leq s$; then $\mathsf{t}_3 = 8$. For $i = 3$, $s \in \mathbf{F}_3$ so that s becomes $10\frac{1}{2}$; thus $\mathsf{t}_2 = 10\frac{1}{2}$.

The problem of computational complexity generally arises from a centralized control in multi-vehicle systems. However,

Algorithm 3 Solution for DEC$(1|r_i, p_i = 1|L_{max}$,0)

procedure POLYNOMIAL$(\mathbf{F}, \mathbf{r}, \mathbf{d})$
  Sort $\mathbf{r}, \mathbf{d}$ into an increasing order of $\mathbf{r}$
  for $i \leftarrow n$ to 1 **do**
    for all j such that $d_j \geq d_i$ **do**
      if $c_j = \emptyset$ **then** $c_j \leftarrow d_j - 1$
      else $c_j \leftarrow c_j - 1$
      if $c_j \in \mathbf{F}_k$ for some k **then**
        $c_j \leftarrow \inf(\mathbf{F}_k)$
    if $i = 1$ or $r_{i-1} < r_i$ **then**
      $c \leftarrow \min_j\{c_j\}$
      if $c < r_i$ **then**
        return $\{\emptyset, no\}$
      else if $r_i \leq c < r_i + 1$ **then**
        $\mathbf{F} \leftarrow \mathbf{F} \cup (c - 1, r_i)$
  for all $i \leftarrow 1$ to n **do**
    if $\nexists r_i$ such that $r_i \leq s$ **then**
      $s \leftarrow \min\{r_i : \mathsf{t}_i$ is empty.$\}$
    while $s \in \mathbf{F}_k$ for some k **do** $s \leftarrow \sup(\mathbf{F}_k)$
    for j such that $d_j = \min_k(d_k)$ where $r_k \leq s$ **do**
      $\mathsf{t}_j \leftarrow s$ and set $s \leftarrow s + 1$
  Sort $\mathsf{T} = \{\mathsf{t}_1, \ldots, \mathsf{t}_n\}$ in the original order
  return $\{\mathsf{T}, yes\}$

notice that Algorithm 3 runs in a polynomial scale with respect to the number of jobs n. We adapt Problem 2 to DEC$(1|r_i, p_i = 1|L_{max}, 0)$ in order to address the complexity problem. To make the constraint $p_i = 1$, we define the maximum process time and normalize the other problem variables in Definition 2. Let

$$\theta_{max} := \max_{i \in \mathcal{C}} \max_{\substack{x_i(0) \in X_i \\ : y_i(0) = \alpha_i}} \{t : y_i(t, u_{i,M}) = \beta_i\} \tag{11}$$

which calculates the worst case of the time to cross an intersection (α_i, β_i) among all controlled vehicles. Considering this as the constant process time, revisit Problem 2.

Problem 4 (Efficient Version of Problem 2): Given an initial condition $\mathbf{x}(0)$, determine whether there exists a schedule $\mathbf{T} \in \mathbb{R}^{n_c}_+$ such that for all $i \in \mathcal{C}$,

$$R_i \leq T_i \leq D_i \tag{12}$$

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

$$T_i \leq T_j \Rightarrow T_i + \theta_{max} \leq T_j \tag{13}$$

$$T_i \leq \bar{R}_\gamma \Rightarrow T_i + \theta_{max} \leq \bar{R}_\gamma \tag{14}$$

$$T_i \geq \bar{R}_\gamma \Rightarrow T_i \geq \bar{P}_\gamma \tag{15}$$

if $T_i = 0$, the conditions (5) and (6) need to be satisfied instead of the conditions (13) and (14).

Notice that (14) and (15) imply that T_i for any $i \in \mathcal{C}$ cannot start during $(\bar{R}_\gamma - \theta_{max}, \bar{P}_\gamma)$ for all $\gamma \in \bar{\mathcal{C}}$. Thus, we declare these time intervals as forbidden region $\mathbf{F}$ from the beginning of Algorithm 3. To simplify notation, we assume, without loss of generality, $\mathcal{C} = \{1, \ldots, n_c\}$ and $\bar{\mathcal{C}} = \{n_c +$

$1, \dots, n\}$. Also assume that $y_i(0) \geq \alpha_i$ for $i \in \{1, \dots, m\}$ and $y_i(0) < \alpha_i$ for $i \in \{m+1, \dots, n_c\}$.

Algorithm 4 Efficient verification of the safety of all vehicles

procedure APPROXSOLUTION($\mathbf{x}(0), \delta$)
  if $\mathbf{y}(0) \in B$ **then**
    return $\{\mathbf{0}, no\}$
  for all $i \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$ **do**
    given $x_i(0)$ calculate R_i, D_i, θ_{max}
    given $x_\gamma(0)$ calculate $\bar{R}_\gamma, \bar{P}_\gamma$
    $\bar{R}_\gamma \leftarrow \max(\bar{R}_\gamma - \delta, 0)$
    $\bar{P}_\gamma \leftarrow \max(\bar{P}_\gamma - \delta, 0)$
  for all $\gamma \in \bar{\mathcal{C}}$ **do**
    $\mathbf{F} \leftarrow \mathbf{F} \cup (\bar{R}_\gamma/\theta_{max} - 1, \bar{P}_\gamma/\theta_{max})$
  for $i \leftarrow 1$ to m **do**
    $T_i \leftarrow 0$
    calculate $P_i(T_i)$ and $P_{max} \leftarrow \max_i P_i(T_i)$
  for $i \leftarrow m+1$ to n_c **do**
    $R_i \leftarrow \max(R_i, P_{max})$
  $\mathbf{r} \leftarrow (R_{m+1}/\theta_{max}, \dots, R_{n_c}/\theta_{max})$
  $\mathbf{d} \leftarrow (D_{m+1}/\theta_{max} + 1, \dots, D_{n_c}/\theta_{max} + 1)$
  $[t_{m+1}, \dots, t_{n_c}, answer] \leftarrow$ POLYNOMIAL($\mathbf{F}, \mathbf{r}, \mathbf{d}$)
  for $i \leftarrow m+1$ to n_c **do** $T_i \leftarrow \theta_{max} t_i$
  return $\{\mathbf{T}, answer\}$

Algorithm 4 is the solution for Problem 4 when $\delta = 0$. Since this algorithm may return *no* even when the solution for Problem 2 exists, we quantify how conservative it is.

Lemma 3: Given T_i and T_j for some $i, j \in \mathcal{C}$ with $y_i(0) < \alpha_i$ and $y_j(0) < \alpha_j$ and given 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 $T_i - T_j < \theta_{max}$ or if $(T_i, T_i + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma) \neq \emptyset$ for some $\gamma \in \bar{\mathcal{C}}$, then there exists $t^* \in [T_i, T_i + \theta_{max}]$ such that $y_j(t^*, u_j) \in (\alpha_j, \alpha_j + \theta_{max}\dot{y}_{j,M})$ or $y_\gamma(t^*, d_\gamma) \in (\alpha_\gamma, \beta_\gamma)$, respectively.

Proof: The first condition $T_i - T_j < \theta_{max}$ implies a violation of (13), which imply collisions between controlled vehicles i and j. To prove this, let $t^* = T_i$. Since $T_j < T_i$, we have $y_j(t^*, u_j) > \alpha_j$. Also, the assumptions that $T_i < T_j + \theta_{max}$ and $\dot{y}_j \leq \dot{y}_{j,M}$ imply $y_j(t^*, u_j) < \alpha_j + \theta_{max}\dot{y}_{j,M}$. Thus, t^* satisfies $y_j(t^*, u_j) \in (\alpha_j, \alpha_j + \theta_{max}\dot{y}_{j,M})$. The second condition does not satisfy (14) and (15), that is, collisions between controlled and uncontrolled vehicles. For an uncontrolled vehicle γ, there is t^* such that $t^* \in (T_i, T_i + \theta_{max}) \cap (\bar{R}_\gamma, \bar{P}_\gamma)$. Since the output is monotone with respect to time, for $t^* > \bar{R}_\gamma$ and $d_\gamma \geq d_{\gamma,m}$, we have $y_\gamma(t^*, d_\gamma) > \alpha_\gamma$; for $t^* < \bar{P}_\gamma$ and $d_\gamma \leq d_{\gamma,M}$, we have $y_\gamma(t^*, d_\gamma) < \beta_\gamma$. Therefore, for some $d_\gamma \in \mathcal{D}_\gamma$, it is shown that $y_\gamma(t^*, d_\gamma) \in (\alpha_\gamma, \beta_\gamma)$. ■

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

$$\hat{B} := \{\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \in Y : y_i \in (\alpha_i, \hat{\beta}_i) \text{ and } y_j \in (\alpha_j, \hat{\beta}_j) \text{ for some } i \neq j \text{ such that } i \in \mathcal{C} \text{ and } j \in \mathcal{C} \cup \bar{\mathcal{C}} \text{ for all } \mathbf{d}\}. \quad (16)$$

The inflated bad set concerns the worst case of the time to cross an intersection.

Theorem 3: Given initial condition $\mathbf{x}(0)$, if APPROXSOLUTION($\mathbf{x}(0), 0$) returns "*no*", then there is no input $\mathbf{u}$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \notin \hat{B}$ for all $\mathbf{d}$ for all $t \geq 0$.

Proof: Lemma 3 indicates that if APPROXSOLUTION($\mathbf{x}(0), 0$) returns *no*, $\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \in \hat{B}$ for any $\mathbf{u}$ for all $\mathbf{d}$. Therefore, this theorem is proven by Lemma 3. ■

We modify Problem 1 by substituting the inflated bad set $\hat{B}$ for the bad set B, that is, the modified verification problem determines whether there exists $\mathbf{u}$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}) \notin \hat{B}$ for all $\mathbf{d} \in \mathcal{D}$ for all $t \geq 0$. Theorem 3 indicates that the modified verification problem and Problem 4 are not equivalent: the necessary condition for the modified verification problem to have such an input $\mathbf{u}$ is that APPROXSOLUTION($\mathbf{x}(0), 0$) returns *yes*.

B. Efficient Supervisor

We now define an efficient supervisor $\hat{s}$ by applying $\hat{B}$ as a bad set instead of B in Problem 3. Also, set an input operator $\hat{\sigma}$ such that $\arg\inf_{u_i \in \mathcal{U}_i}\{t \geq 0 : y_i(t, u_i) = \hat{\beta}_i\}$ for $i \in \mathcal{C}$ with the same constraint in (9). Moreover, define s_{approx} by substituting APPROXSOLUTION for EXACTSOLUTION in Algorithm 2. The supervisor s_{approx} is not equivalent to $\hat{s}$ in the same sense of Theorem 3. Thus we provide that s_{approx} is less conservative than or equally conservative to $\hat{s}$. To prove this, we first investigate the relation between APPROXSOLUTION and EXACTSOLUTION. Then, we verify the non-blocking property of s_{approx}.

Lemma 4: Given a measured state $\mathbf{x}(k\tau)$, if APPROXSOLUTION($\mathbf{x}(k\tau), 0$) returns *yes*, then EXACTSOLUTION($\mathbf{x}(k\tau), 0$) also returns *yes*.

Proof: Notice that condition (12) and (15) are identical to condition (4) and (7), respectively. From the definition of θ_{max} in (11), for any $i \in \mathcal{C}$ for any T_i, $T_i + \theta_{max} \geq P_i(T_i)$. Thus, condition (13) and (14) become $P_i(T_i) \leq T_i + \theta_{max} \leq T_j$ and $P_i(T_i) \leq T_i + \theta_{max} \leq \bar{R}_\gamma$, satisfying condition (5) and (6). ■

Lemma 5: If APPROXSOLUTION($\mathbf{x}(k\tau), 0$) $= \{\mathbf{T}, yes\}$, then $\mathbf{u} := \hat{\sigma}(\mathbf{x}(k\tau), \mathbf{T}) \neq \emptyset$, then for $\mathbf{u}_k$ which restricts $\mathbf{u}$ to $[k\tau, (k+1)\tau]$, APPROXSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_k), \tau$) $= \{\mathbf{T}, yes\}$.

Proof: From Lemma 4, the assumption that APPROXSOLUTION($\mathbf{x}(k\tau), 0$) returns *yes* implies EXACTSOLUTION($\mathbf{x}(k\tau), 0$) returns *yes*. Then, by the first part of Lemma 1, $\sigma(\mathbf{x}(k\tau), \mathbf{T})$ exists so that $\hat{\sigma}(\mathbf{x}(k\tau), \mathbf{T})$ is non-empty. Let $\mathcal{C} = \{1, \dots, n_c\}$ and $\bar{\mathcal{C}} = \{n_c + 1, \dots, n\}$. For the same notation in Algorithm 4, suppose $y_i(0) \geq \alpha_i$ for $i \in \{1, \dots, m\}$. For $\mathbf{x}(k\tau)$, there exists a feasible schedule $\{T_{m+1}, \dots, T_{n_c}\}$ satisfying $|T_i - T_j| \geq \theta_{max}$ and $T_i \notin \mathbf{F}_k\theta_{max}$ for all $i, j \in \{m+1, \dots, n_c\}$. In order for APPROXSOLUTION($\mathbf{x}_{k+1}(\mathbf{u}_k), \tau$) to return *yes*, POLYNOMIAL($\mathbf{F}_{k+1}, \mathbf{r}_{k+1}, \mathbf{d}_{k+1}$) must admit a feasible schedule where $\mathbf{F}_{k+1}, \mathbf{r}_{k+1}, \mathbf{d}_{k+1}$ represent an idle-time, release time, and due date at time $t = (k+1)\tau$. Let $\tilde{\mathbf{u}}$ restrict $\mathbf{u}$ to $[(k+1)\tau, \infty)$. Since $\mathbf{u} = \tilde{\mathbf{u}} \cup \mathbf{u}_k$, $\mathbf{x}_{k+1}(\mathbf{u}_k)$ have a schedule T_i' such that $T_i' = T_i - \tau \in [R_i', D_i']$ by construction.

This schedule satisfies both $|T'_i - T'_j| \geq \theta_{max}$ and $T'_i \notin \mathbf{F}_{k+1}\theta_{max}$ resulting from the time delay $\delta = \tau$ so that $\mathbf{F}_{k+1}\theta_{max} = \mathbf{F}_k\theta_{max} - \tau$. Therefore a feasible schedule in terms of $\mathbf{x}_{k+1}(\mathbf{u}_k)$ exists. ■

Lemma 6: If APPROXSOLUTION$(\mathbf{x}_{k+1}(\mathbf{a}_k), \tau) = \{\mathbf{T}, yes\}$, then $\hat{\sigma}(\mathbf{x}_{k+1}(\mathbf{a}_k), \mathbf{T}) \neq \emptyset$.

Proof: As in the proof of Lemma 2, set $\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k)$ for any $\mathbf{d}_k$. The corresponding forbidden region is denoted by $\tilde{\mathbf{F}}_{k+1}$. Since $\mathbf{F}_{k+1}$, defined from $\mathbf{x}_{k+1}(\mathbf{a}_k)$, considers all possible idle-times, we have $\tilde{\mathbf{F}}_{k+1}\theta_{max} \subseteq \mathbf{F}_{k+1}\theta_{max}$. The schedule $\mathbf{T} = \{T_{m+1}, \ldots, T_{n_c}\}$ for APPROXSOLUTION$(\mathbf{x}_{k+1}(\mathbf{a}_k), \tau)$ satisfies $T_i \in [R_i, D_i]$,$|T_i - T_j| \geq \theta_{max}$ and $T_i \notin \mathbf{F}_{k+1}\theta_{max}$. This schedule also satisfies $T_i \notin \tilde{\mathbf{F}}_{k+1}\theta_{max}$; thus, APPROXSOLUTION$(\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k), 0)$ returns *yes*. From the first part of Lemma 5, we have a non-empty $\hat{\sigma}(\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k), \mathbf{T})$. Since $\hat{\sigma}$ is the composition of $\hat{\sigma}_i$ for $i \in \mathcal{C}$, and $\tilde{\mathbf{x}}_{k+1}(\mathbf{a}_k)$ is equivalent to $\mathbf{x}_{k+1}(\mathbf{a}_k)$ for $i \in \mathcal{C}$, we conclude $\sigma(\mathbf{x}_{k+1}(\mathbf{a}_k), \mathbf{T}) \neq \emptyset$. ■

Theorem 4: The supervisor $s_{approx}(\mathbf{x}(k\tau), \mathbf{a}_k)$ is no more restrictive than $\hat{s}(\mathbf{x}(k\tau), \mathbf{a}_k)$ and non-blocking in the sense described in Problem 3.

Proof: Theorem 3 implies that APPROXSOLUTION returns yes as long as an input profile exists to keep away from $\hat{B}$. Therefore, $s_{approx}(\mathbf{x}(k\tau), \mathbf{a}_k)$ is less restrictive than $\hat{s}(\mathbf{x}(k\tau), \mathbf{a}_k)$. The non-blocking property follows from Lemma 5 and 6 with the same argument in the proof of Theorem 2. ■

VI. NUMERICAL SIMULATIONS

We test the supervisory algorithm in Section IV and V. Consider 6 controlled and 2 uncontrolled vehicles whose dynamical state is $x_i = [p_i; v_i]$. With an input u_i and a disturbance d_γ for $i \in \mathcal{C}$ and $\gamma \in \bar{\mathcal{C}}$, each set of vehicles is modelled as follows.

$$\dot{p}_i = v_i \qquad \dot{v}_i = u_i$$
$$\dot{p}_\gamma = v_\gamma \qquad \dot{v}_\gamma = d_\gamma.$$

VII. CONCLUSIONS

In this paper, we have designed a supervisor that prevents an intersection collision among all vehicles in the presence of uncontrolled vehicles. The design of a supervisor is based on two main problems: verification of the safety of all vehicles, and management of the inputs of controlled vehicles. An inserted idle-time (IIT) scheduling is applied to determine future safety even though there are vehicles uncontrolled. We have proved that the supervisor has a non-blocking algorithm so that it is least restrictive, that is, it intervenes in a situation only if necessary. Since the problem of computational complexity arises in a centralized control like a supervisor, we use an efficient version of scheduling so that address the problem from an inflated bad set.

The assumption that zero speed is not considered in the design of the supervisor increases traffic flow and energy efficiency. While the presence of uncontrolled vehicles is in the context of practical situations, several real-world issues still exist. For future work, the case that vehicles have multi-directional paths should be considered. As noted, the uncertainties of measurement and process are considered in [6], and rear-end collisions are considered in [5].

REFERENCES

[1] J. J. Kanet and V. Sridharan, "Scheduling with inserted idle time: Problem taxonomy and literature review," *Operations Research*, vol. 48, no. 1, pp. 99–110, Jan. 2000.

[2] "National motor vehicle crash causation survey, u.s. department of transportation," http://www-nrd.nhtsa.dot.gov/Pubs/811059.pdf.

[3] "U.s. department of transportation,its strategic research plan 2010-2014," http://www.its.dot.gov/strategic_plan2010_2014/, 2008.

[4] M. R. Hafner and D. Del Vecchio, "Computational tools for the safety control of a class of piecewise continuous systems with imperfect information on a partial order," *SIAM Journal on Control and Optimization*, vol. 49, no. 6, pp. 2463–2493, Jan. 2011.

[5] A. Colombo and D. Del Vecchio, "Cooperative conflict resolution:a scheduling approach," *IEEE Transactions on Automatic Control*.

[6] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling." IEEE Conference on Decision and Control.

[7] M. Heymann, F. Lin, and G. Meyer, "Control synthesis for a class of hybrid systems subject to configuration-based safety constraints," in *Hybrid and Real-Time Systems*, ser. Lecture Notes in Computer Science, O. Maler, Ed. Springer Berlin Heidelberg, Jan. 1997, no. 1201, pp. 376–390.

[8] L. Bakule, "Decentralized control: An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87–98, Apr. 2008.

[9] M. R. Garey, D. S. Johnson, B. B. Simons, and R. E. Tarjan, "Scheduling UnitTime tasks with arbitrary release times and deadlines," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 256–269, May 1981.

[10] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*. Courier Dover Publications, 2003.

[11] B. Simons, "A fast algorithm for single processor scheduling," in , *19th Annual Symposium on Foundations of Computer Science, 1978*, 1978, pp. 246–252.

[12] J. Lenstra, A. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," in *Annals of Discrete Mathematics*, E. J. P.L. Hammer, Ed. Elsevier, 1977, vol. Volume 1, pp. 343–362.

[13] T. H. Cormen, *Introduction To Algorithms*. MIT Press, 2001.

[14] B. A. Davey, *Introduction to lattices and order*. Cambridge university press, 2002.

[15] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: ACM, 2012, p. 145154.