Safety Verification Methods for Human-Driven Vehicles at Traffic Intersections: Optimal Driver-Adaptive Supervisory Control

Gabriel Rodrigues de Campos, Fabio Della Rossa, and Alessandro Colombo

***Abstract*—We design an optimal, driver-adaptive supervisor for collision avoidance at an intersection. The algorithm is able to identify optimal corrections to the human-decided inputs and to keep the system collision free. To determine the set of safe control actions, we exploit the notion of maximal controlled invariant set. We leverage results from scheduling theory to verify the safety of a given control input, and propose an efficient optimization algorithm providing optimal solutions with respect to the drivers' intent. We also present an approximate supervisor algorithm that can be solved in polynomial time and has guaranteed error bounds. Finally, we validate our approach with simulation results, as well as on naturalistic data.**

***Index Terms*—Collision avoidance, intersection, safety verification algorithms, supervisory control, traffic coordination.**

I. Introduction

MODERN transportation systems are increasingly relying on communication technologies and automatic control [1], [2], and a particular area of interest of recent research on smart mobility is intersection management. Even if intersections represent a small part of the entire road system, they account for a significant part of traffic accidents. According to recent reports, 20% and 21.5% of traffic fatalities during the last decade are intersection related within the EU and the USA, respectively, [3], [4]. Most important, close to 94% of accidents are completely, or in part, due to human error as a result of misinterpretation of a situation, inattention or the disregard of traffic rules [3], [5]. Such alarming numbers justify the design of increasingly sophisticated semiautonomous and autonomous safety systems, aimed to provide more efficient, comfortable, and almost accident-free road traffic.

In this paper, we focus our attention on collision avoidance algorithms at traffic intersections. We assume that humans are driving each car, and that a (centralized or decentralized) supervisor is in charge of ensuring the vehicles' safety (i.e., a human-in-the-loop system, see Fig. 1). We consider that the information on the environment and surrounding vehicles and an estimation on the drivers' intent are available and exchanged via a wireless network between the vehicles. We abstract from the perception and drivers' intent estimation problems, proposing a generic algorithm that can cope with *any* intent estimation algorithm, e.g., [6]–[8]. We aim at designing a driver-adaptive supervisor (blue element of Fig. 1). In practice, such a supervisor can be implemented in two fundamentally different frameworks. In a *centralized* framework, a single supervisor functions as the process manager, and the control inputs are communicated over wireless links. This setup requires two communication hops: one for exchanging the state information and a second for the control policy. In a *decentralized* framework instead, we consider that computation is performed by supervisor units installed in each vehicle, where each supervisor has access to global vehicles' information transmitted over omnidirectional communication links. The reader can refer to [9] for a detailed discussion on decentralized and distributed sensing and control.

The problem of supervision for collision avoidance is discussed, among others, in [10]–[18], and is typically set in a framework of verification for safety specifications. Though standard general purpose algorithms exist, they are limited by

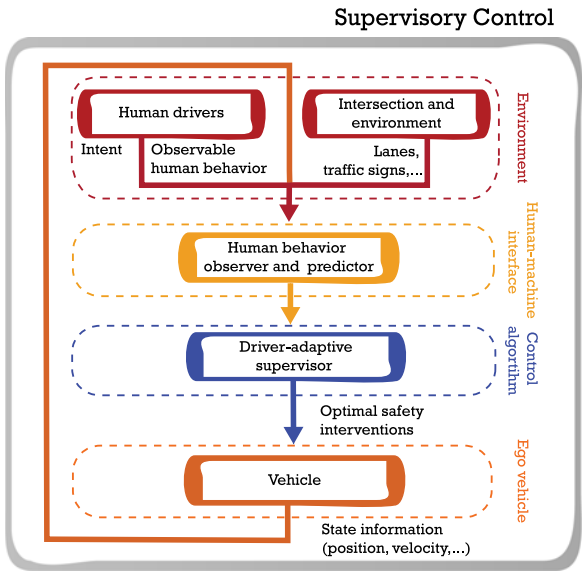

Fig. 1. Supervisory control structure of human-driven vehicles.

Manuscript received April 19, 2016; revised October 28, 2016, May 2, 2017, and August 27, 2017; accepted October 25, 2017. This work was supported under Grant "AD14VARI02—Progetto ERC BETTER CARS—Sottomisura B." This paper was recommended by Associate Editor Dr. Laura Stanley. *(Corresponding author: Gabriel Rodrigues de Campos.)*

The authors are with the DEIB, Politecnico di Milano, Milano 20133, Italy (e-mail: gabriel@decampos.eu; fabio.dellarossa@polimi.it; alessandro.colombo@polimi.it).

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

Digital Object Identifier 10.1109/THMS.2017.2776205

numerical complexity to handle problems involving just a few agents (typically two). A set of efficient solutions for the intersection collision avoidance problem was proposed in [13] using the scheduling theory, and extended to more complex scenarios in [15]–[18]. Note that all the aforementioned papers focus solely on the safety aspects, and ignore in their design optimality arguments: no attempt is done to approximate the drivers' intent when the drivers' input is overridden. Hence, there may be a mismatch between the input returned by the supervisor and the drivers' desired input. To cope with this problem, we propose an optimal, driver-adaptive solution. Our approach is based on the solution of two separate problems: 1) The *verification problem* (VP), determining if there exists an input signal that leads all agents safely through the intersection, and 2) the *supervisor problem*, returning a safe and optimal approximation of the drivers' intent if the desired input violates safety conditions. To determine the set of safe control actions, we exploit the notion of maximal controlled invariant set (MCIS). Even if our approach is aimed at human-driven vehicles, it can be coupled with existing algorithms for autonomous vehicles [19]–[24], in a multilayer control structure. In this case, our algorithm would ensure the safety of the trajectories generated by a higher level decision system.

The contributions of this paper are the following. First, we elaborate on a novel optimal conflict resolution technique first presented in [25] that is optimal with respect to the human's desired actions. Second, we propose an optimal supervisor coupled with a state predictor robust to input uncertainties, easily extendable to also handle modeling and measurement uncertainties. Third, we discuss here approximate solutions that guarantee real-time implementation of the proposed solution for set of more than eight vehicles. Finally, we validate our theoretical framework not only on simulated but also on real data. Note that, though building on our previous work [25], all the theoretical machinery (definitions, lemmas, theorems, ...) is novel due to the robust state prediction and the definition of the MCIS given later. This paper is organized as follows. Section II describes the dynamic model, and Section III the problem formulation. The different steps of our approach are presented subsequently: the state prediction in Section IV; safety verification in Section V; and control synthesis in Section VI. The properties of our supervisor algorithm are discussed in Section VII, simulations and experimental results given in Section VIII, and our conclusions provided in Section IX.

II. System Definition

Consider the system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),\ \mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in X \subseteq \mathbb{R}^{rn}$ is the state of n vehicles moving on n different paths with r-order dynamics, $\mathbf{y} \in \mathbb{R}^n$ is the vector of the positions of the vehicles along their paths, and $\mathbf{u}$ is a vector of control inputs. The system is given by the parallel composition of n different systems

$$\dot{x}_i = f_i(x_i, u_i),\ y_i = h_i(x_i) \tag{2}$$

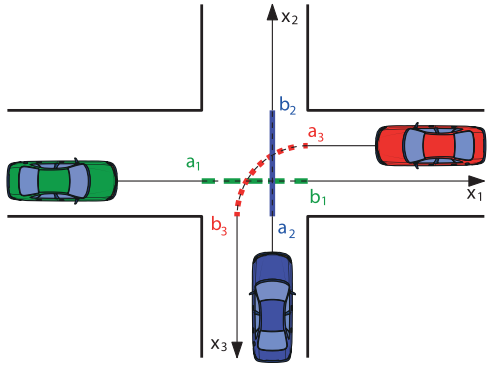

Fig. 2. Illustration of the considered scenario. Several human-driven vehicles approach an intersection following predefined paths.

describing the longitudinal dynamics of each vehicle. We assume that system (1) has unique solutions and that the individual systems (2) are monotone [26], with $\mathbb{R}_+$ (the nonnegative real line) as the positivity cone of y_i. This yields

$$\begin{array}{c}(x_i(0), \dot{x}_i(0)) \succeq (x_i'(0), \dot{x}_i'(0)), u_i \succeq u_i' \\ \Downarrow \\ (x_i, \dot{x}_i) \succeq (x_i', \dot{x}_i').\end{array} \tag{3}$$

In words, this means that the more a vehicle accelerates, the faster it will move. Throughout this paper, x_i, y_i, and u_i will be used indifferently to denote vectors (as above) and signals, the correct interpretation will be clear from the context. The values of $\mathbf{x}$ and $\mathbf{y}$ at time t, starting from $\mathbf{x}_0$ and with input signals $\mathbf{u}$, are denoted $\mathbf{x}(t, \mathbf{u}, \mathbf{x}_0)$ and $\mathbf{y}(t, \mathbf{u}, \mathbf{x}_0)$. The functional spaces of the input signals $u_i(t)$ and $\mathbf{u}$ are $\mathcal{U}_i$ and $\mathcal{U} \subset \mathbb{R}^n$, respectively, and the set $\mathcal{U}_i$ is compact, with a unique maximum $u_{i,\max}$ and minimum $u_{i,\min}$. We also assume that $\dot{y}_i$ is bounded to the nonnegative interval $[0, \dot{y}_{i,\max}]$ for all i and that $\lim_{t\to\infty} \dot{y}_i(t, u_{i,\max}) = \dot{y}_{i,\max}$. Without loss of generality, and for simplifying the notation, we assume that vehicles have homogeneous input sets $\mathcal{U}_i$, such that $u_{i,\min} = u_{j,\min} = u_{\min}$ and $u_{i,\max} = u_{j,\max} = u_{\max}, \forall i, j$.

In this paper, we assume that the path of each vehicle i is known, and that vehicles do not change paths/lanes when they engage in the intersection. The intersection can be modeled as an interval (a_i, b_i) along each path, see Fig. 2. Note that the interval (a_i, b_i) should be defined in such a way that the size of vehicles and the intersection itself are accounted for. Let $B \subset \mathbb{R}^n$ denote the (time invariant) *bad set*, including all configurations corresponding to a side collision, i.e., all y_i and y_j such that $y_i(t) \in (a_i, b_i)$ and $y_j(t) \in (a_j, b_j)$ at the same instant t

$$B := \{\mathbf{y} \in \mathbb{R}^n : y_i \in (a_i, b_i) \wedge y_j \in (a_j, b_j), \text{for some } i \neq j\}.$$

The supervisor that we present later on is implemented as a discrete-time algorithm. To keep notation simple, we consider in the rest of this paper that 0 is the current time when a step of the supervisor is executed. Finally, we also introduce the following definition.

Definition 1 (Notation convention): We denote the following:

1) $\mathbf{u}_m$ the measurement of brake and acceleration input for all drivers, taken at time zero and from time zero onwards;

2) $\mathbf{u}_{\text{des}}(t)$ the unknown, future desired control signal for all drivers;
3) $\mathbf{u}_{\text{hyp}}(t)$ a hypothesis on the future behavior of the drivers, i.e., a hypothesis on $\mathbf{u}_{\text{des}}(t)$;
4) $\mathbf{u}_{\text{meas}}(t)$ a constant input signal, equal to $\mathbf{u}_m$.

III. Problem Statement

We assume that humans are driving each car, and that a supervisor is in charge of verifying the safety of human-decided control inputs [18], [27]. In the literature, the subset of X of all initial conditions admitting a safe input is known as the MCIS, see [28]. Define $[\mathbf{x}^l, \mathbf{x}^h] := \{\mathbf{x} : \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^h\}$, where $\mathbf{x}^l$ and $\mathbf{x}^h$ represent a lower and upper bound on the state vector, respectively. Formally, the MCIS is given as follows.

Definition 2: The set $[\mathbf{x}^l, \mathbf{x}^h] \subseteq X$ belongs to the MCIS if and only if there exists $\mathbf{u} \in \mathcal{U}$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{x}_0) \notin B$ for all $t \geq 0$ and for all $\mathbf{x}_0 \in [\mathbf{x}^l, \mathbf{x}^h]$.

Note that the definition above still holds when the set of states is a singleton. As long as the system's state remains within the MCIS, there exists an input that avoids collisions. Therefore, the role of the supervisor is to ensure that the state never leaves the MCIS, while modifying the input selected by the drivers as little as possible.

Let $\mathbf{u} : \mathbb{R}_+ \to \mathbb{R}^n$ be the input returned by the supervisor. The *supervisor problem* can be posed as the following.

Problem 1 (Supervisor Problem): Given the current state $\mathbf{x}_0$, the input signal $\mathbf{u}_{\text{meas}}$ and a cost function $J(\mathbf{u}_{\text{meas}}, \mathbf{u})$, return $\mathbf{u}$ such that the future state will not enter the bad set B, and so as to minimize $J(\mathbf{u}_{\text{meas}}, \mathbf{u})$.

Here, asking for the future state not to enter the bad set B, reads $f(\mathbf{x}_0, \mathbf{u}) \in \mathcal{T}_{\mathbf{x}}$ MCIS $\forall t$, where $\mathcal{T}_{\mathbf{x}}$ MCIS denotes the *tangent cone* at $\mathbf{x}$ to the MCIS and corresponds to the set of all vectors $\mathbf{v}$ such that

$$\lim_{\mathbf{x}_k \to \mathbf{x}, \mathbf{x}_k, \mathbf{x} \in \text{MCIS}} \frac{\mathbf{x}_k - \mathbf{x}}{\|\mathbf{x}_k - \mathbf{x}\|} = \frac{\mathbf{v}}{\|\mathbf{v}\|}.$$

The above limit corresponds to the set of infinitesimal perturbations $(\mathbf{x}_k - \mathbf{x})$ such that $\mathbf{x}_k \in$ MCIS. The supervisor routine is then composed of the following sequential steps.

1) *State prediction* over a horizon θ given $\mathbf{u}_{\text{hyp}}$.
2) *Safety verification*, ensuring that the set of predicted states can be reached without collisions using $\mathbf{u}_{\text{hyp}}$, and that there exists an input that avoids collisions for all $t > \theta$, for all states in the set.
3) *Control synthesis*, when the safety verification fails.

The above supervisor is implemented as a discrete-time algorithm, with a fixed time stepping τ (larger than the worst case computational time). An illustration of this principle is given in Fig. 3. We will now analyze the three steps separately.

IV. State Prediction

Even if the driver's high-level objectives (e.g., go straight or turn at the intersection) are assumed to be known *a priori* inferred by a high-level intent identification algorithm [6], [29], the supervisor can only measure the current brake and acceleration input, $\mathbf{u}_m$ (see Definition 1). In the impossibility of exactly knowing the drivers' future desired control signal $\mathbf{u}_{\text{des}}$, safety

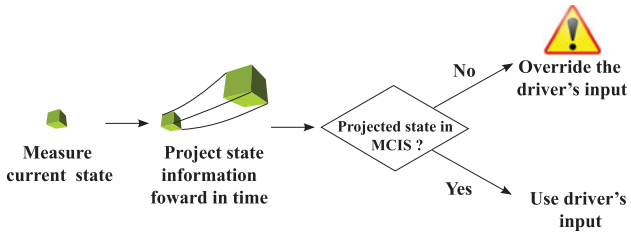

Fig. 3. Illustration of the working principle of the proposed supervisory control. Given the estimated driver intent and the current measurement of the system's state, a state prediction over a predefined horizon is performed. For the resulting state predictions, the verification problem (VP) is solved: if the verification succeeds, the driver's input is returned; otherwise, a safe control input needs to be computed.

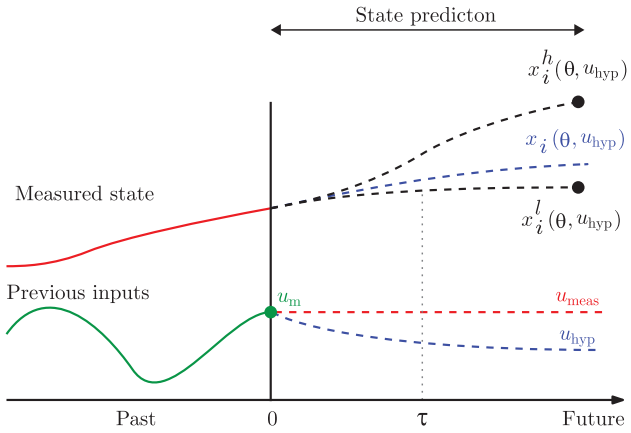

Fig. 4. Notation convention: illustration according to Definition 1.

systems aiming to optimize a car's response around $\mathbf{u}_{\text{des}}$ should therefore incorporate a suitable inference algorithm for the signal $\mathbf{u}_{\text{des}}$. In this paper, we do not focus on the way to infer a hypothesis $\mathbf{u}_{\text{hyp}}$, but we define our state predictor so that the resulting architecture correctly works for any possible hypothesis.

Let θ be the prediction horizon and $\mathbf{u}_{\text{hyp}}(t) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}]$ represent *any* hypothesis on the driver's behavior. Recall that 0 is considered to be the current time at which the supervisor problem is solved. The lower- and upper-bounds of the estimated trajectory are defined as follows:

$$\mathbf{x}^l(t, \mathbf{u}_{\text{hyp}}) := \mathbf{x}(t, \mathbf{u}, \mathbf{x}_0) \text{ s.t. } \begin{cases} \mathbf{u} = \mathbf{u}_{\min}, & \text{for } t \in [0, \tau] \\ \mathbf{u} = \mathbf{u}_{\text{hyp}}(t), & \text{for } t \in (\tau, \theta] \end{cases} \tag{4}$$

$$\mathbf{x}^h(t, \mathbf{u}_{\text{hyp}}) := \mathbf{x}(t, \mathbf{u}, \mathbf{x}_0) \text{ s.t. } \begin{cases} \mathbf{u} = \mathbf{u}_{\max}, & \text{for } t \in [0, \tau] \\ \mathbf{u} = \mathbf{u}_{\text{hyp}}(t), & \text{for } t \in (\tau, \theta] \end{cases} \tag{5}$$

See Fig. 4 for an illustration. The following lemma holds as a consequence of monotonicity.

Lemma 1: Given the current state measurement $\mathbf{x}_0$, $\mathbf{x}^l(t, \mathbf{u}_{\text{hyp}}) \leq \mathbf{x}(t, \mathbf{u}, \mathbf{x}_0) \leq \mathbf{x}^h(t, \mathbf{u}_{\text{hyp}})$ for all $0 \leq t \leq \tau$ and for all $\mathbf{u}(t) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}]$.

Here, we relax the assumptions on the driver's behavior of [25], where the driver's input was considered to be a fixed, constant signal: (4) and (5) consider a control input equal to $\mathbf{u}_{\min}$

and $\mathbf{u}_{\max}$ for $t \in [0, \tau]$, respectively, and $\mathbf{u}_{\text{hyp}} \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}]$ for $t \in (\tau, \theta]$. This is important to guarantee that the supervisor proposed later is nonblocking, a property that would not be satisfied if the state predictions were solely based on $\mathbf{u}_{\text{hyp}}(t)$.

V. Safety Verification

In order to guarantee safety, collisions should be avoided for all future times. Hence, guaranteeing safety comes to ensuring that the set of (infinite horizon) control actions avoiding all conflicts is nonempty. Consider a prediction horizon θ and an expected control signal $\mathbf{u}_{\text{hyp}}(t)$. We formally define the VP as follows.

Problem 2 (VP): Given the set of state estimations $[\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})]$, determine if there exists an input signal $\mathbf{u}$ which guarantees that $\mathbf{y}(t, \mathbf{u}, \mathbf{x}_0) \notin B$ for all $t \geq 0$ and for all $\mathbf{x}_0 \in [\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})]$.

In other words, we need to verify that

$$[\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})] \in \text{MCIS}. \tag{6}$$

To verify the above condition, we exploit the representation of the constraint (6) in terms of a scheduling problem, following the idea introduced in [13]. We briefly introduce this equivalence in the following section.

A. Equivalence Between Verification Problem and Scheduling Problem

Let $\mathbf{y}^l(t, \mathbf{u}) = h(\mathbf{x}^l(t, \mathbf{u}))$ and $\mathbf{y}^h(t, \mathbf{u}) = h(\mathbf{x}^h(t, \mathbf{u}))$. Define for each agent i with $y_i(0) \leq a_i$ the quantities $R_i := \min\{t \geq 0 : y_i^h(t, u_{\max}) \geq a_i\}$, $D_i := \min\{t \geq 0 : y_i^h(t, u_{\min}) \geq a_i\}$. These two quantities are, respectively, the minimum and maximum time at which $y_i^h(t, u_i)$ reaches the intersection (and 0 if $y_i^h(t, u_i) > a_i$). Notice that R_i is always finite, since by assumption $\lim_{t\to\infty} y_i(t, u_{i,\max}) = \dot{y}_{i,\max}$, while D_i can in general be infinite if $u_{i,\min}$ can bring agent i to a stop before a_i. For each agent i with $y_i^h(0) \leq a_i$, given a real number T_i, define $P_i(T_i) := \min_{u_i \in \mathcal{U}_i}\{t : y_i^l(t, u_i) = b_i\}$, with constraint $y_i^h(t, u_i) \leq a_i \ \forall\, t < T_i$. If the constraint cannot be satisfied, set $P_i(T_i) := \infty$. If $[y_i^l(0), y_i^h(0)] \cap (a_i, b_i) \neq \emptyset$ define $P_i(T_i) := \min\{t : y_i^l(t, u_{i,\max}) = b_i\}$, and if $y_i^l(0) \geq b_i$ define $P_i(T_i) := 0$. $P_i(T_i)$ is the earliest time that y_i^l can reach b_i, if y_i^h does not pass a_i before T_i.

A scheduling problem consists in assigning jobs to a resource satisfying given requirements [30]. Using the above quantities, we can write the VP as a scheduling problem where the intersection represents the resource, the agents represent the job to be assigned to the resource, and the time spent by each agent in the intersection is the length of the job to be executed. The following result holds.

Theorem 1: The interval $[\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})] \in \text{MCIS}$ if and only if there exists a schedule $\mathbf{T} = (T_1, \ldots, T_n) \in \mathbb{R}_+^n$ such that for all i

$$R_i \leq T_i \leq D_i \tag{7}$$

$$T_i \geq T_j \Rightarrow T_i \geq P_j(T_j). \tag{8}$$

The proof follows directly from [15], and will be omitted here. In accordance with the results of Theorem 1, the solution to (6) can be found using Algorithm 1. Given the set of initial conditions $[\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})]$, Algorithm 1 calculates $\mathbf{R} = [R_1, \ldots, R_n]$ and $\mathbf{D} = [D_1, \ldots, D_n]$ and retrieves a schedule (if one exists) by testing all the possible ordering permutations of n agents.

Algorithm 1: [T, answer] = ExactVP ($\mathbf{x}$, $\mathbf{u}_m$).

```
x_0 ← x
u_hyp(t) ← f(u_m) ∀t ≥ 0
for all i ∈ {1, ..., n} do
    given [x^l(θ, u_hyp), x^h(θ, u_hyp)] calculate R_i and D_i
end for
for all permutations of {1, ..., n} do
    T_1 ← R_1
    for i ∈ {2, ..., n} do
        T_i ← max(P_{i-1}(T_{i-1}), R_i)
    end for
    if T_i ≤ D_i for all i ∈ {1, ..., n} then
        return {T, yes}
    end if
end for
return {∅, no}
```

There is also an extension of the scheduling problem , defined by (7) and (8), where jobs cannot be executed during specified time intervals. These are known as *inserted idle times* (iit), see [16]. This particular type of scheduling problem will be necessary later for the derivation of a multiobjective optimization algorithm.

VI. Control Synthesis

When safety verification fails [according to (6)], the last stage of the supervisor routine requires the synthesis of a safe control signal minimizing a given performance metric. Before introducing the proposed optimization algorithm, let us discuss the optimization objectives and the implementation aspects of the supervisor algorithm. The supervisor routine is implemented with a time stepping τ, meaning that the generated output is the optimal input signal $\mathbf{u}$ for the interval $[0, \tau]$, i.e., until the next instant when the supervisor routine is performed. Since the only information available from the drivers is the current input $\mathbf{u}_m$, a sensible objective is to minimize the difference between the supervisor correction and the constant signal $\mathbf{u}_{\text{meas}}(t)$, equal to $\mathbf{u}_m$ for all $t \geq 0$. This choice is suitable for the chosen numerical method given in the sequel, which requests a constant signal, and approximates well the drivers' intent as long as τ is small with respect to the drivers' input rate of change. We will therefore define the cost function as the infinity norm of the difference between the supervisor output and $\mathbf{u}_{\text{meas}}$

$$J(\mathbf{u}_{\text{meas}}, \mathbf{u}) := \|\mathbf{u} - \mathbf{u}_{\text{meas}}\|_\infty. \tag{9}$$

By using the infinity norm, we minimize the worst case difference between u_i and $u_{\text{meas},i}$ for all vehicles. We can then formalize the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u}\in\mathcal{U}} \quad & J(\mathbf{u}_{\text{meas}}, \mathbf{u}) \\ \text{subject to} \quad & \mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS} \end{aligned} \tag{10}$$

where the optimization constraint is expressed as the solution of the VP, and can be addressed using the techniques discussed in [13] and [18], which exploit a similar equivalence to the one presented in Section V-A. In the sequel, we provide a numerical strategy to solve problem (10).

Remark 1: Note that this approach differs from previous works in the domain, which ignore in their design any optimality arguments. In the nonoptimized implementation of the supervisor problem presented in [13] and [15]–[18], no attempt is done to approximate the drivers' intent when the drivers' input is overridden. Hence, this is equivalent to solving (10) with the cost function

$$J(\mathbf{u}_{\text{des}}, \mathbf{u}) := \begin{cases} 0, & \text{if } \mathbf{u}_{\text{meas}} = \mathbf{u} \\ 1, & \text{if } \mathbf{u}_{\text{meas}} \neq \mathbf{u}. \end{cases} \tag{11}$$

This corresponds to returning $\mathbf{u}_{\text{meas}}$ whenever this maintains the state within the MCIS, and to returning an arbitrary input $\mathbf{u}$ such that $\mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}$ otherwise. Though effective, this can lead to unwanted or unnecessarily aggressive decelerations/accelerations.

A. Single Objective Control Design

Let u_{bound} be an upper bound to (9). Problem (10) can then be reformulated as

$$\begin{aligned} \min_{\mathbf{u} \in \mathcal{U}} \quad & u_{\text{bound}} \\ \text{subject to} \quad & \|\mathbf{u} - \mathbf{u}_{\text{meas}}\|_\infty \leq u_{\text{bound}} \\ & \mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}. \end{aligned} \tag{12}$$

Clearly, problems (10) and (12) are equivalent: the solution of (12) minimizing $\mathbf{u}_{\text{bound}}$ corresponds to the optimal solution of (10). However, the search space of (12) is a functional space. To simplify the optimization problem, define $\text{MCIS}(u_{\text{bound}})$ as the set of all states $\mathbf{x} \in X$ satisfying the VP under the constraint

$$\|\mathbf{u}(t) - \mathbf{u}_{\text{meas}}(t)\|_\infty \leq u_{\text{bound}}, \text{ for } t \in [0, \theta]. \tag{13}$$

Then, we can write

$$\begin{aligned} \min_{u_{\text{bound}} \in \mathbb{R}_+} \quad & u_{\text{bound}} \\ \text{subject to} \quad & \mathbf{x}_0 \in \text{MCIS}(u_{\text{bound}}) \end{aligned} \tag{14}$$

where the search space is now the nonnegative real line. The following result holds.

Lemma 2: The optimal cost of (12) is equal to the optimal cost of (14).

Proof: Let u'^*_{bound} and u^*_{bound} represent the optimal costs of (12) and (14), respectively. The following two arguments hold.

1) $u^*_{\text{bound}} \geq u'^*_{\text{bound}}$: If $\mathbf{x}_0 \in \text{MCIS}(u^*_{\text{bound}})$, then there exists $\bar{\mathbf{u}}$ such that $\|\bar{\mathbf{u}} - \mathbf{u}_{\text{des}}\|_\infty \leq u^*_{\text{bound}}$ for all $t \in [0, \theta]$ and $\mathbf{x}(\theta, \bar{\mathbf{u}}, \mathbf{x}_0) \in \text{MCIS}$. Take now $\mathbf{u} = \bar{\mathbf{u}}$ for $t \in [0, \theta]$ and $\mathbf{u} = \mathbf{u}_{\text{des}}$ for $t > \theta$. This gives $\|\mathbf{u} - \mathbf{u}_{\text{des}}\|_\infty \leq u^*_{\text{bound}}$ and $\mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}$. Thus, $(\bar{\mathbf{u}}, u^*_{\text{bound}})$ is a feasible solution for (12).
2) $u'^*_{\text{bound}} \geq u^*_{\text{bound}}$: If $\mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}$ and $\|\mathbf{u} - \mathbf{u}_{\text{des}}\|_\infty \leq u'^*_{\text{bound}}$, then $\mathbf{x}_0 \in \text{MCIS}(u'^*_{\text{bound}})$. Therefore, u'^*_{bound} is a feasible solution for (14).

Because of the previous two statements, $u^*_{\text{bound}} = u'^*_{\text{bound}}$. ■

Algorithm 2: Numerical solution of (14).

```
Initialise U = max_i(u_{i,max} − u_{i,min}), L = 0
while U − L > threshold do
    u_bound = (U + L)/2
    if x_0 ∈ MCIS(u_bound) then
        U = u_bound
    else
        L = u_bound
    end if
end while
```

The optimal cost u^*_{bound} of (12) is the smallest value of the cost function (9) for which all agents can avoid collisions. The optimal solution u^*_{bound} to (14) can be numerically computed using the bisection method (see Algorithm 2), and an optimal solution of (12) retrieved by selecting an input $\mathbf{u}$ satisfying the constraints of (12) for $u_{\text{bound}} = u^*_{\text{bound}}$. Ways to construct such an input are explained in [13], for example.

Note that there can be multiple optimal solutions $\mathbf{u}$ with the same cost u^*_{bound}: an emergency manoeuvre necessary to avoid a collision between two vehicles may set a large u^*_{bound}, hindering the optimization of the supervisor correction for a third vehicle unaffected by the collision. This is particularly clear in the results presented in Section VIII-A. In other words, there is a set of optimal solutions to problem (10), and the single-agent cost functions $J_i(u_{\text{meas},i}, u_i)$ induce a preorder on this set. In the next section, we explore the solution structure of (12) in terms of Pareto optimality. We formulate the supervisor problem as a multiobjective optimization problem and show how to retrieve an optimal solution.

Remark 2: Algorithm 2 inherits the complexity of the verification step $\mathbf{x}_0 \in \text{MCIS}(u_{\text{bound}})$, since the bisection loop is $O(1)$. Therefore, the complexity of optimally solving (10) is comparable to that of solving the VP.

Remark 3: The optimality of problem (12) is not dependent on $\mathbf{u}_{\text{hyp}}$. A less accurate hypothesis $\mathbf{u}_{\text{hyp}}$ will only increase the number of interventions (i.e., the total time the supervisor overrides the driver's input). Once overriding, the optimization procedure is independent of $\mathbf{u}_{\text{hyp}}$ and it is shown here to be Pareto optimal. Recall that safety is always guaranteed due to the robust state prediction for $t \in [0, \tau]$.

B. Multiobjective Control Design

Rewrite problem (10) as the following multiobjective optimization problem:

$$\begin{aligned} \min_{u_1 \in \mathcal{U}_1} \quad & J_1(u_{\text{meas},1}, u_1) \\ \min_{u_2 \in \mathcal{U}_2} \quad & J_2(u_{\text{meas},2}, u_2) \\ & \vdots \\ \min_{u_n \in \mathcal{U}_n} \quad & J_n(u_{\text{meas},n}, u_n) \\ \text{subject to} \quad & \mathbf{x}(\theta, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}. \end{aligned} \tag{15}$$

We introduce the following definition.

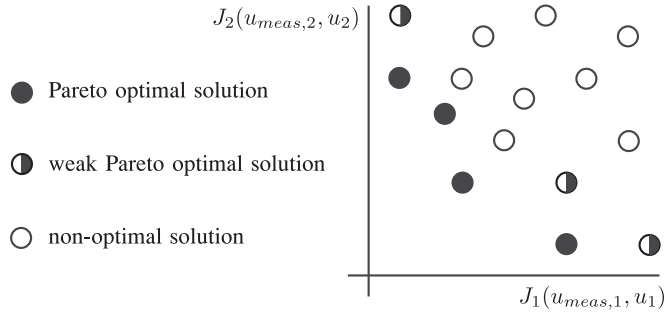

Fig. 5. Illustration of Pareto and weak-Pareto solutions, according to Definition 3.

Definition 3: An admissible solution $\mathbf{u}$ of (15) is called *weak Pareto optimal* if there exists no admissible solution $\mathbf{u}'$ such that $J_i(u_{\text{meas},i}, u_i') < J_i(u_{\text{meas},i}, u_i)$ for all i; among the weak Pareto optimal solutions, $\mathbf{u}$ is called *Pareto optimal* if there exists no admissible solution $\mathbf{u}' \neq \mathbf{u}$ such that: 1) $J_i(u_{\text{meas},i}, u_i') \leq J_i(u_{\text{meas},i}, u_i)$ for all i, and 2) $J_i(u_{\text{meas},i}, u_i') < J_i(u_{\text{meas},i}, u_i)$ for at least one i. An illustration is presented in Fig. 5.

It follows from Definition 3 that Pareto optimal solutions are not comparable in the preorder induced by (15), i.e., all Pareto optimal solutions are equally good. Considering Definition 3, we introduce our next result.

Lemma 3: All optimal solutions of (10) are weak Pareto Optimal for (15).

Proof: We prove this result by contradiction. Assume that there is an optimal solution $\mathbf{u}$ of (10) that is not Pareto optimal for (15). This means that there exists a solution $\mathbf{u}'$ of (15) such that $J_i(u_{\text{meas},i}, u_i') < J_i(u_{meas,i}, u_i)$ for all i. Then, $J(\mathbf{u}_{\text{meas}}, \mathbf{u}') < J(\mathbf{u}_{\text{meas}}, \mathbf{u})$ in (10), which contradicts the optimality of $\mathbf{u}$. ■

By the above lemma, any optimal solution of (10) is at least weak Pareto optimal. Nevertheless, our ultimate goal is to select, among all optimal solutions, one that is Pareto optimal.

To find the optimal solution to (15), we will exploit in the sequel the equivalence described in Section V-A. Recall now the definitions of the quantities R_i, D_i, and P_i and note that they are all dependent on the set $\mathcal{U}_i$. In the presence of constraint (13), such quantities become a function of the constraining quantity u_{bound}. Hence, we define $R_i(u_{\text{bound}})$, $D_i(u_{\text{bound}})$, and $P_i(T_i, u_{\text{bound}})$ as in Theorem 1, with the additional constraint $\|\mathbf{u}(t) - \mathbf{u}_{\text{meas}}(t)\|_\infty \leq u_{\text{bound}}$ for $t \in [0, \theta]$. We introduce the following definition.

Definition 4 (Scheduling Problems): Letting SP denote a scheduling problem defined by (7) and (8), we write the following.

1) $\text{SP}(u_{\text{bound}})$ when the scheduling quantities are computed under the constraint (13).
2) $\text{SP}(u_{1,\text{bound}}, \ldots, u_{n,\text{bound}})$ when the constraint (13) is different for different agents.
3) $\mathbf{T} \in \text{SP}(u_{\text{bound}})$ if $\mathbf{T}$ is a feasible schedule of $\text{SP}(u_{\text{bound}})$.
4) $\text{SP}(L, u_{\text{bound}})$ when a restriction of $\text{SP}(u_{\text{bound}})$ to a subset L of the agents $\{1, \ldots, n\}$ is considered.
5) $\text{SP}(u_{\text{bound}}, IIT)$ when an additional constraint

$$(T_i, P_i(T_i)) \cap (\alpha_j,\ \beta_j) = \emptyset \ \forall\, i, j \neq i. \tag{16}$$

is added to $\text{SP}(u_{\text{bound}})$, given a set of inserted idle times IIT:= $\{[\alpha_1, \beta_1], [\alpha_2, \beta_2], \ldots\}$.

6) $\text{SP}(L, u_{\text{bound}}, \text{IIT})$ when constraint (16) is added to $\text{SP}(L, u_{\text{bound}})$, given a set of inserted idle times IIT:= $\{[\alpha_1, \beta_1], [\alpha_2, \beta_2], \ldots\}$.

Using this new notation and Theorem 1, problem (14) can be rewritten as

$$\begin{aligned} \min_{u_{\text{bound}} \in \mathbb{R}_+} \quad & u_{\text{bound}} \\ \text{subject to} \quad & \exists \mathbf{T} : \mathbf{T} \in \text{SP}(u_{\text{bound}}). \end{aligned} \tag{17}$$

The following holds.

Lemma 4: Consider the quantities R_i, D_i, and $P_i(T_i)$ of $\text{SP}(u_{\text{bound}})$, and R_i', D_i', and $P_i'(T_i)$ of $\text{SP}(u_{\text{bound}}')$ with $u_{\text{bound}}' < u_{\text{bound}}$. We have that $R_i \leq R_i'$, $P_i(T_i) \leq P_i'(T_i)$, $D_i \geq D_i'$.

Proof: The property follows from the fact that $\text{SP}(u_{\text{bound}})$ is a relaxation of $\text{SP}(u_{\text{bound}}')$. Note that changing the value of u_{bound} is equivalent to changing the bounds of the feasible set of inputs $\mathcal{U}_i$. Hence, $\mathcal{U}_i(u_{\text{bound}}') \subseteq \mathcal{U}_i(u_{\text{bound}})$ whenever $u_{\text{bound}}' < u_{\text{bound}}$, and since R_i', D_i', and $P_i'(T_i)$ are defined for the extremal points of $\mathcal{U}_i(u_{\text{bound}}')$, the previous result holds. ■

It follows from the previous result that by decreasing the value of u_{bound} one tightens the constraints of $\text{SP}(u_{\text{bound}})$. As a consequence, we can interpret the optimal cost of (12) as the value u_{bound}^* for which a subset of jobs verifies the constraints exactly, i.e., would not be schedulable for a smaller value of u_{bound}. Based on this interpretation, we introduce the following definition.

Definition 5 (Tight set): Consider a schedule $\mathbf{T} \in \text{SP}(u_{\text{bound}}, \text{IIT})$. We say that an ordered set of jobs and inserted idle times $i \in \{1, \ldots, m\}$ is tight if the following conditions are satisfied.

1) All jobs and iit's except the first start exactly after the previous job or iit is done, i.e., $T_i = P_{i-1}(T_{i-1})$, or $T_i = \beta_{i-1}$, or $\alpha_i = P_{i-1}(T_{i-1})$, or $\alpha_i = \beta_{i-1}$.
2) If the first element is a job it starts exactly at its release time, i.e., at R_1.
3) If the last element is a job, it starts exactly at its deadline D_m.

In words, a tight set is a set of jobs and iit's whose scheduled starting time cannot be changed without changing the order in which they are executed. Note that a single job with equal release time and deadline is a minimal example of a tight set, and that an iit is by definition always a minimal tight set.

Given a tight set for a schedule $\mathbf{T} \in \text{SP}(u_{\text{bound}}, \text{IIT})$, let *constrained* jobs denote the subset of jobs which do not satisfy constraints (7), (8), or (16) if u_{bound} is reduced, unless we change the order with which they are scheduled in $\mathbf{T}$. Let $P_i(u_{\text{bound}})$ and $D_i(u_{\text{bound}})$ denote the scheduling quantities of problem $\text{SP}(u_{\text{bound}})$ with dependence on u_{bound}. The following definition is introduced.

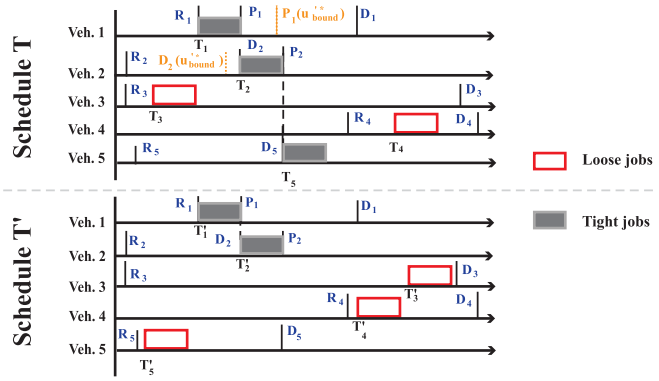

Fig. 6. Illustration of two feasible schedules for $\mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$. Jobs 1 and 2 are constrained in both schedules, whereas job 5 is constraining in T and not tight in T'. Schedule T' is then the constraint-minimal between the two.

Definition 6 (Constrained and constraining jobs): A tight job i is *constrained* in a schedule $\mathbf{T}$ for a problem $\mathrm{SP}(u_{\mathrm{bound}}, IIT)$ if the following performances hold:

1) it is followed by another tight job j and $P_i(T_i, u'_{\mathrm{bound}}) > T_j$ for any $u'_{\mathrm{bound}} < u_{\mathrm{bound}}$; or
2) it is followed by an IIT $[\alpha, \beta]$ and $P_i(T_i, u'_{\mathrm{bound}}) > \alpha$ for any $u'_{\mathrm{bound}} < u_{\mathrm{bound}}$; or
3) $T_i > D_i(u'_{\mathrm{bound}})$ for any $u'_{\mathrm{bound}} < u_{\mathrm{bound}}$.

A tight job is *constraining* if it is not constrained and it is preceded by a constrained job in the same set of tight jobs.

Hence, one can think of the constraining jobs for a schedule $\mathbf{T} \in \mathrm{SP}(u_{\mathrm{bound}}, \mathrm{IIT})$ as those jobs which limit the minimum value u_{bound} can take while allowing $\mathbf{T}$ to be adapted to be feasible in $\mathrm{SP}(u_{\mathrm{bound}}, \mathrm{IIT})$, without changing the relative order of jobs and iit's. This leads us to the concept of a *constraint-minimal schedule*, defined as follows.

Definition 7 (Constraint-minimal schedule): Consider a schedule $\mathbf{T} \in \mathrm{SP}$. The schedule is constraint-minimal if no other schedule $\mathbf{T}' \neq \mathbf{T}$, $\mathbf{T}' \in \mathrm{SP}$ has a set of constrained jobs that is a strict subset of that of $\mathbf{T}$.

An illustrative example of constrained/constraining jobs and constraint-minimal schedules is given in Fig. 6.

From Lemma 4, it follows that u^*_{bound} defines the subset of jobs that verify the scheduling constraints exactly, i.e., would not be schedulable for a smaller u_{bound} values. Hence, u^*_{bound} corresponds to the worst case scenario imposed by this set of jobs, as the remaining jobs could still be schedulable for lower values of u^*_{bound}. The proposed solution to find an optimal value of $u^*_{\mathrm{bound},i}$ is based in the following construction.

Procedure 1: (Reduction step)

1) Consider a schedule $\mathbf{T} \in \mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$, where u^*_{bound} is the optimal cost of (12) with constraint $\exists\ \mathbf{T}$: $\mathbf{T} \in \mathrm{SP}(u_{\mathrm{bound}}, \mathrm{IIT})$, and assume that $\mathbf{T}$ is constraint-minimal.
2) Define a set C of constrained jobs and L of jobs that are not constrained in $\mathbf{T}$ for $\mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$, and define a new set $\mathrm{IIT}' := \mathrm{IIT} \cup \{[T_i, P_i(T_i)] \forall i \in C\}$.
3) Call u'^*_{bound} the optimal cost of (14) with constraint $\exists\ \mathbf{T}'$: $\mathbf{T}' \in \mathrm{SP}(L, u_{\mathrm{bound}}, \mathrm{IIT}')$.
4) Finally, consider the scheduling problem $\mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$, where $u''^*_{\mathrm{bound},i} := u^*_{\mathrm{bound}}$ if $i \in C$, and $u''^*_{\mathrm{bound},i} := u'^*_{\mathrm{bound}}$ if $i \in L$.

In other words, we suggest to identify the subset of constrained jobs verifying the scheduling constraints exactly with u^*_{bound}, and remove them from the optimization problem (12) by reserving their execution time as iit. This lead us to the following result.

Lemma 5: The set of constrained jobs in $\mathbf{T}$ for $\mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$ is a subset of the set of constrained jobs in any $\mathbf{T}'' \in \mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$ for $\mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$.

Proof: This is a consequence of selecting a constraint-minimal schedule $\mathbf{T}$. First of all, notice that: 1) $u'^*_{\mathrm{bound}} < u^*_{\mathrm{bound}}$, by Lemma 4 and since u'^*_{bound} is computed by removing from SP all constraining jobs, and 2) for any $\mathbf{T}'' \in \mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$, $\mathbf{T}'' \in \mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$. The only way that a job which is constrained in $\mathbf{T}$ for $\mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$ can be not constrained in $\mathbf{T}''$ for $\mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$, is if its constraining job j is scheduled at a different time in $\mathbf{T}''$ than in $\mathbf{T}$. However, since $\mathbf{T}$ is constraint-minimal, scheduling j at any different time would generate a new constrained job k in $\mathbf{T}''$ for $\mathrm{SP}(u^*_{\mathrm{bound}}, \mathrm{IIT})$. We have defined $u''^*_{\mathrm{bound},k} = u'^*_{\mathrm{bound}} < u^*_{\mathrm{bound}}$, therefore this would imply $\mathbf{T}'' \notin \mathrm{SP}(u''^*_{\mathrm{bound},1}, \ldots, u''^*_{\mathrm{bound},n}, \mathrm{IIT})$. ■

By iterating the reasoning of Procedure 1, it is then possible to retrieve a Pareto optimal solution to problem (15). The structure of the proposed multiobjective optimization procedure is presented in Algorithm 3, and leads to the next result.

Theorem 2: Algorithm 3 provides a Pareto optimal solution to (15).

Proof: The algorithm is implementing the process described before Lemma 5, and returns a schedule $\mathbf{T}^*$ and a solution to (15) in terms of a set of optimal costs $J^*_1, \ldots, J^*_n$. From Lemma 5, we can conclude that all jobs for the schedule $\mathbf{T}^* \in \mathrm{SP}(J^*_1, \ldots, J^*_n)$ are constrained, or have $J^*_i = 0$. In both cases, it is not possible to find a feasible schedule for a problem $\mathrm{SP}(J'^*_1, \ldots, J'^*_n)$ with $J'^*_i < J^*_i$ for at least one i, therefore $J^*_1, \ldots, J^*_n$ is a Pareto optimal solution. ■

Remark 4: The complexity of Algorithm 3 is defined by the complexity of the optimization step, which in turn is determined by the complexity of the test at line 4 in Algorithm 2.

The outcome of the multiobjective optimization algorithm and the advantages with respect to the single-objective problem are discussed later in Section VIII-A, where some simulation examples are presented.

VII. Supervisory Control

In this section, we present a supervisor algorithm that solves Problem 1, by sequentially performing the state prediction, the safety verification, and the control synthesis stages. The supervisor overrides the driver's input signal if and only if the safety

Algorithm 3: $[\mathbf{u}^*_{\text{bound}}, \mathbf{T}^*] = \text{MultiObjOpt}(\mathbf{x}, \mathbf{u}_m)$.

1: $\mathbf{x_0} \leftarrow \mathbf{x}$
2: $\mathbf{u}_{\text{meas}}(t) \leftarrow \mathbf{u}_m \ \ \forall t \geq 0$
3: Initialise $\mathbf{L} = \{1, \ldots, n\}$, IIT $= \emptyset$, $U^* = \infty$
4: set $k = 0$
5: **while** $\mathbf{L}$ is nonempty or $U^* > 0$ **do**
6: $k = k + 1$
7: **Optimization step:** Solve problem(17): $\min u_{\text{bound}}$
8: subject to $\exists \mathbf{T} : \mathbf{T} \in \text{SP}(L, u_{\text{bound}}, IIT)$
9: and call U^* its optimal cost
10: **Selection step:** Select a schedule
$\mathbf{T}^k \in \text{SP}(L, U^*, IIT)$
11: that is constraint-minimal
12: **Reduction step:**
13: **for all** jobs $i \in \mathbf{L}$ that are constrained for $\mathbf{T}^k$ **do**
14: remove i from $\mathbf{L}$
15: add the interval $[T_i, P_i(T_i)]$ to IIT
16: set $u^*_{\text{bound},i} := U^*$, $T^*_i := T^k_i$
17: **end for**
18: **end while**
19: **return** $(u^*_{\text{bound},1}, \ldots, u^*_{\text{bound},n})$, $(T^*_1, \ldots, T^*_n)$

Algorithm 4: Supervisor$(\mathbf{x}, \mathbf{u}_m)$.

1: $\mathbf{u}_{\text{hyp}}(t) \leftarrow f(\mathbf{u}_m) \ \ \forall t \geq 0$
2: $\mathbf{u}_{\text{meas}}(t) \leftarrow \mathbf{u}_m \ \ \forall t \geq 0$
3: $\{\mathbf{T}, \textit{answer}\} \leftarrow$ ExactVP$(\mathbf{x_0}, \mathbf{u}_{\text{hyp}})$
4: **if** *answer* $=$ *yes* **then**
5: leave the drivers do whatever they want. **return**
6: **else**
7: $\{\mathbf{u}^*_{\text{bound}}, \mathbf{T}^*\} \leftarrow$ MultiObjOpt $(\mathbf{x_0}, \mathbf{u}_{\text{meas}})$
8: override the driver input using $\mathbf{u}_{\text{opt}}$ defined in (18). **return**
9: **end if**

verification stage fails, i.e., whenever

$$\nexists\, \mathbf{u} \in \mathcal{U} \text{ s.t. } \mathbf{y}(t, \mathbf{u}, \mathbf{x}_0) \cap B = \emptyset \ \forall t \geq 0$$
$$\forall \mathbf{x_0} \in [\mathbf{x}^l(\theta, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\theta, \mathbf{u}_{\text{hyp}})]$$

for all the agents i for which $y_i(0) \leq b_i$. Recall that $\mathbf{u}_{\text{hyp}}$ is defined in Definition 1. In such cases, the overriding signal $\mathbf{u}_{\text{opt}}$ is given as

$$u_{\text{opt},i} := \arg \inf_{u_i \in \mathcal{U}_i} \left\{ t \geq 0 : y^l_i(t, u_i) \geq b_i \right\}$$
$$\text{with constraints: } \|u_i - u_{\text{meas},i}\|_\infty \leq u^*_{\text{bound},i}$$
$$y^h_i(t, u_i) \leq a_i \text{ for } t < T^*_i \tag{18}$$

where $(u^*_{\text{bound},i}, T^*_i)$ are given by Algorithm 3. In words, the supervisor defines, when needed, an input $u_{\text{opt},i}(t)$ allowing agent i to exit the intersection no later than $t = P_i(T^*_i)$ or to enter it before T^*_i, while satisfying input constraints $u^*_{\text{bound},i}$. The structure of the proposed supervisor algorithm is given in Algorithm 4. The following result holds.

Theorem 3: Assume that $\mathbf{x}_0 \in$ MCIS. Then, Algorithm 4 1) solves the supervisor problem and 2) is nonblocking.

Proof:

1) To prove (i), consider that Algorithm 4 returns $\mathbf{u}_{\text{des}}$ as long as a schedule exists satisfying Theorem 1, i.e., as long as $[\mathbf{y}^l(t, \mathbf{u}_{\text{hyp}}), \mathbf{y}^h(t, \mathbf{u}_{\text{hyp}})] \cap B = \emptyset, \ \forall t \geq 0$. If that is not the case, condition (6) is not satisfied and an (override) input u_{opt} is returned, which is defined according to Algorithm 3. As such input is safe and has been derived s.t. it minimizes $J(\mathbf{u}_{\text{meas}}, \mathbf{u})$, this concludes the proof.
2) Let $\mathbf{u}$ be the supervisor output at time $t = 0$. From Lemma 1, it follows that $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in [\mathbf{x}^l(\tau, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\tau, \mathbf{u}_{\text{hyp}})], \ \forall \ \mathbf{u}$. As shown before, the proposed algorithm correctly solves the supervisor problem, which means that $[\mathbf{y}^l(t, \mathbf{u}_{\text{hyp}}), \mathbf{y}^h(t, \mathbf{u}_{\text{hyp}})] \cap B = \emptyset, \ \forall t \geq 0$. Hence, this yields that $\forall \ \mathbf{x}(\tau) \in [\mathbf{x}^l(\tau, \mathbf{u}_{\text{hyp}}), \mathbf{x}^h(\tau, \mathbf{u}_{\text{hyp}})]$, there exists at least an $\bar{\mathbf{u}}$ s.t. $[\mathbf{y}^l(t, \bar{\mathbf{u}}), \mathbf{y}^h(t, \bar{\mathbf{u}})] \cap B = \emptyset, \ \forall t \geq \tau$, where $\bar{\mathbf{u}}$ is the supervisor output at time $t = \tau$. Therefore, the admissible set of the supervisor at time τ is nonempty, i.e., the supervisor is nonblocking. ■

Remark 5: The previous results can be extended in order to also cope with measurement noise. Using an approach similar to [15], the proposed algorithm can be reformulated based on the computation of the maximal robust controlled invariant set, i.e., the largest set of states of inputs that avoids conflicts for all positive times and for any admissible disturbance. By leveraging the monotonicity properties and the uniform continuity of the system's flow, one can derive a robust supervisor algorithm.

A. Approximate Supervisor

In the previous section, we provided an algorithm that determines exactly the membership in the MCIS according to (6). But this verification is often a computationally difficult problem, and has been proved to be NP-hard for some collision avoidance problems of practical interest [13], [31]. A number of exact algorithms have been proposed, whose application to systems with more than a few agents is not practical [10]–[12], [14]. Some of these results are applicable only to the two-agent conflict resolution problem, and the others have exponential complexity in the size of the state space.

Nevertheless, approximations exist with polynomially bounded running time. An obvious way to reduce the computation time of Algorithm 3 to polynomial is to solve a scheduling problem $\text{SP}^{\text{fixed-order}}$ with a predetermined job order instead of SP at lines 3 and 3 of Algorithm 3. This, however, does not guarantee in general an error bound. To choose a job order with a guaranteed error bound, we can preprocess jobs using the ideas presented in [13], [15], and [18]: we define a time $\delta_{\max}(u_{\text{bound}})$, which is long enough so that any agent can cross the interval (a_i, b_i) in at most $\delta_{\max}(u_{\text{bound}})$, and allocate this fixed amount of time to each agent. Define $\text{SP}^{\text{fixed-length}}$ as the scheduling problem with jobs of equal length $\delta_{\max}(u_{\text{bound}})$. By substituting SP by $\text{SP}^{\text{fixed-length}}$ at lines 3 and 3 of Algorithm 3, we can exploit the polynomial-time scheduling algorithm proposed in [32] (and reported as algorithm POLYNOMIALTIME in [13]) to compute an optimal schedule, i.e., an optimal job order.

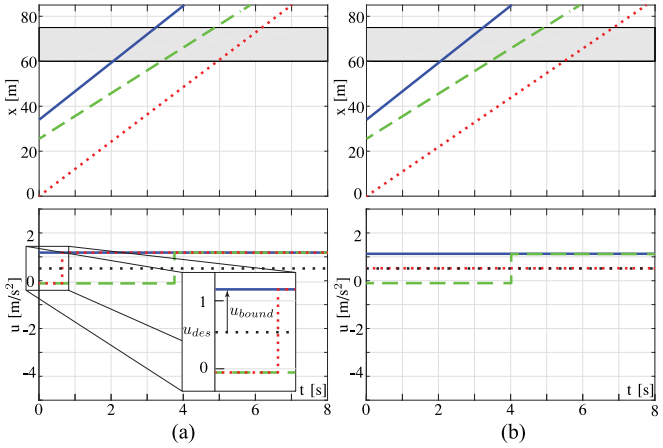

Fig. 7. Single-objective versus multiobjective optimization: (a) Single-objective; (b) Multiobjective optimization with $\theta = 5$ and $u_{\text{hyp}} = u_{\text{meas}} = 0.5$ (black dotted line).

Note that the solution found is a Pareto optimal solution among all schedules with job lengths $\delta_{\max}(u_{\text{bound}})$. This optimal order is then fixed and used in one more run of Algorithm 3 with $\text{SP}^{\text{fixed-order}}$.

In the next section, we will present simulation results highlighting the advantages of this approximate algorithm.

VIII. Results

We consider in the sequel a multivehicle scenario as depicted in Fig. 2. We assume that all agents are moving over different paths and that their longitudinal dynamics are described by double integrator dynamics given by

$$\ddot{x}_i(t) = u_i(t),\ y_i(t) = x_i(t) \tag{19}$$

where $\dot{x}_i \in [0 \text{ m/s}, 17 \text{ m/s}]$ and $u_i \in [-5 \text{ m/s}^2, 3 \text{ m/s}^2]$, $\forall i$. Note that a linear model has been chosen here for the sake of simplicity. Nevertheless, the results of this paper also hold for nonlinear dynamical models that satisfy the monotonicity properties mentioned in Section III. With the exception of Figs. 8 and 11, each subfigure is composed of two panels. In the top panel, the intersection is represented by a grey box, and the position trajectories of the different vehicles are in color. In the lower panel, $\mathbf{u}$ and $\mathbf{u}_{\text{meas}}$ are represented by solid and dotted lines, respectively. The following results were obtained on a 2.8 GHz, 16 Gb RAM laptop with Windows 10, using MATLAB 2016b.

A. Single-Objective versus Multiobjective Optimization

We analyze in this section, the performances of the single- and multiobjective algorithms. For this scenario, the initial conditions of the system are $\mathbf{x} = [(0, 10), (24, 10), (32, 10)]$ and the prediction horizon is $\theta = 5$ s. To simplify the interpretation of the results, we assume that the drivers of all vehicles always request an input equal to $u_{\text{hyp}} = u_{\text{meas}} = 0.5$ (horizontal dotted line in the bottom panels of Fig. 7), and that the intersection corresponds to the interval $[60, 75]$ m along all vehicles' path.

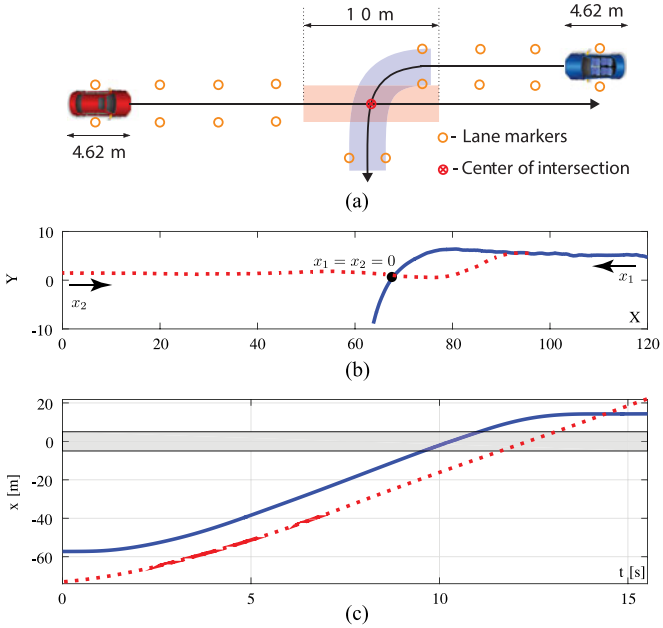

Fig. 8. Experimental setup for data collection: (a) Illustration of the intersection scenario; (b) Naturalistic trajectories of the two vehicles; (c) Normalized trajectories, where the intersection is set to be at the origin with a length equal to twice the size of the vehicles. X and Y are local coordinates.

Fig. 7(a) shows the result of the single-objective optimization problem (14), for which the optimal solution is $u^*_{\text{bound}} = 0.53$. One can see that, to avoid a collision between the blue (solid line) and green (dashed line) vehicle, the optimal control policy forces all agents to deviate from their desired control input. All vehicles apply a control signal where the maximum difference with respect to u_{meas} corresponds to u^*_{bound}, see the zoom on the lower image of Fig. 7(a). Note, however, that the red vehicle (dotted line) is not involved in an immediate collision with the remaining vehicles and there is no reason to alter its trajectory.

Fig. 7(b) shows the result of the multiobjective optimization problem (15). As expected, the performance of the optimization algorithm improves. More precisely, only the blue (solid line) and green (dashed line) vehicles' trajectories are corrected, allowing the red vehicle (dotted line) to continue its desired trajectory. Without needing to correct unnecessarily the red vehicle, the multiobjective optimization algorithm is, as expected, less restrictive.

B. Naturalistic Data Validation

We validate now the performances of our control algorithms with naturalistic data, using two Volvo S60 T6 vehicles. We considered an intersection scenario, as illustrated in Fig. 8(a), and performed multiple tests in velocity ranges going from 20 to 40 km/h. For positioning, each vehicle was equipped with inertial measurement and differential global positioning system modules OXTS RT2002. Vehicles were used to generate realistic trajectories at intersections, which were later fed as user-desired inputs to our supervisor algorithm (i.e., there is no online su-

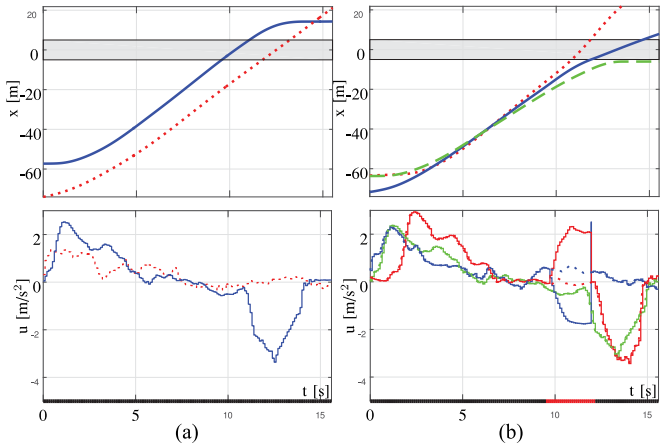

Fig. 9. Supervisor Algorithm 4 applied to a real set of data with $\tau = 0.2$ s and $\theta = 1$ s: (a) Collision-free case; (b) Collision case. In the bottom panels, the dotted curves represent the measured input, while the continuous curves correspond to the input given by the supervisor. The time axis is red if the supervisor is overriding the desired input, black otherwise.

pervisor controlling the vehicles). The GPS-based trajectories are presented in Fig. 8(b). We defined the intersection as an interval centered around coordinate 0 on each path. The length of the each car is 4.62 m and the intersection width is 10 m. The distanced travelled by each vehicle along its own path is given in Fig. 8(c).

In Fig. 9, we used the vehicle model (19) with the driver's input $\mathbf{u}_m$ being equal to the second derivative of the trajectories of Fig. 8(c) and of an additional third trajectory. We assume that $\mathbf{u}_{\rm hyp}$ is a constant signal equal to the last measurement $\mathbf{u}_m$. In other words, $\mathbf{u}_{\rm hyp} = \mathbf{u}_{\rm meas}$, see Definition 1. In the figure, $\mathbf{u}_{\rm meas}$ is illustrated as a dotted curve of color corresponding to the color of each vehicle. The dotted curve is only visible when $\mathbf{u}_m$ is different than the input returned by the supervisor. The supervisor runs with a time stepping of $\tau = 0.1$ s and we define $\theta = 1$ s. The time interval where the supervisor is overriding the drivers' input is highlighted as the red portion of the time axis in the bottom panel of Fig. 9(b).

In Fig. 9(a), where the vehicles perform a safe manoeuvre, we can see that the supervisor never overrides the drivers: the hypothesis on the driver's behavior $\mathbf{u}_{\rm hyp}$ is always safe according to (6). Such results show that, given the expected input signal $\mathbf{u}_{\rm hyp}$, the proposed supervisor does not unnecessarily override the proposed control input and leaves the control of the vehicle to the drivers whenever their behavior is considered safe. Fig. 9(b) considers a different dataset of three vehicles. Here, the vehicles' inputs have been (artificially) shifted in time so that it is coherent with a collision between the blue (solid line) and red (dotted line) vehicles, while the green vehicle (dashed line) performs a safe manoeuver: considering the collision threat, its stops before reaching the intersection. As expected, the proposed supervisor is able to identify vehicles that need to be overridden from those that do not: the supervisor only overrides the blue (solid line) and red (dotted line) vehicles from $t = 9.4$ s until $t = 12.5$ s, see the red portion of the x-axis Fig. 9(b). During this interval, the blue (solid line) vehicle is forced to accelerate while the red (dotted line) decelerates, and this while minimizing the infinity norm error with respect to the input provided by the drivers. However, the green (dashed line) vehicle is never overridden (it is behaving safely), even though $\mathbf{u}_{\rm hyp}$ is different from the driver's desired input. For this scenario, the maximum time to run the optimization algorithm was 0.03 s.

C. Simulation Results

In the following, we present simulation results for a three-vehicle system. In all simulations, the initial conditions of the system are $\mathbf{x} = [(0, 10), (8, 10), (16, 10)]$ and the supervisor runs with a time stepping of $\tau = 0.1$ s. For simplicity, we assume that the drivers of all vehicles always request an input equal to $\mathbf{u}_{\rm meas} = 1$, and that the intersection corresponds to the interval $[60, 75]$ m along all vehicles' path. Furthermore, in order to discuss later the influence of the prediction horizon θ, we consider four distinct values $\theta \in \{0.1, 0.2, 1, 2\}$ s. We consider the nonoptimized approach given in [18] for all vehicle i in the interval $[a_i - 30, b_i]$ m. Vehicles outside this interval never need an override (they can stop before the intersection or they have passed it), and can therefore be removed from the supervisor problem.

The nonoptimized supervisor solution is presented in Fig. 10(a), for $\theta = 0.1$ s. We will use this case to highlight the advantages of the proposed optimal design. One can see that at $t = 3.2$ s into the simulation, the supervisor detects that vehicles are about to leave the MCIS and intervenes by applying bang–bang control inputs.

Fig. 10(b) shows the behavior of the proposed optimal supervisor when $\theta = 0.1$ s. See that the time instant when the supervisor intervenes for the first time is identical to Fig. 10(a). However, the control input profiles differ. While in Fig. 10(a), the supervisor immediately proposes bang–bang inputs, in Fig. 10(b) the control inputs are optimized, see the "stair-like" profile of the green (dashed) and red (dotted) curves before $t = 4$ s. Our solution is able to identify vehicles that need to be overridden from those that do not, see the red (dotted line) trajectory, which is overridden slightly later than in Fig. 10(a). This effect in magnified in Fig. 7.

Fig. 10(c) considers the same scenario when $\theta = 0.2$ s. By optimizing over a longer prediction horizon, it provides a better approximation of the drivers' inputs while avoiding two consecutive collisions: a three-vehicle conflict from $t = 3$ s to $t = 4.6$ s and a two vehicle's conflict from $t = 4.6$ s to $t = 6.3$ s.

Finally, Fig. 10(d) and (e) consider the cases where $\theta = 1$ s and $\theta = 2$ s, respectively. When compared to the previous cases, we can see that this leads to more driver friendly, less aggressive manoeuvres. Generally speaking, there is a tradeoff on the restrictiveness of the supervisor: as the value of θ increases, interventions will be triggered earlier than strictly necessary. This is clear if one compares the red segment of the horizontal axis (which represents the period during which the supervisor intervenes) between Fig. 10(d) and (e).

We also perform randomized simulations for a set of 10 000 initial conditions. We consider a three vehicle setup re-

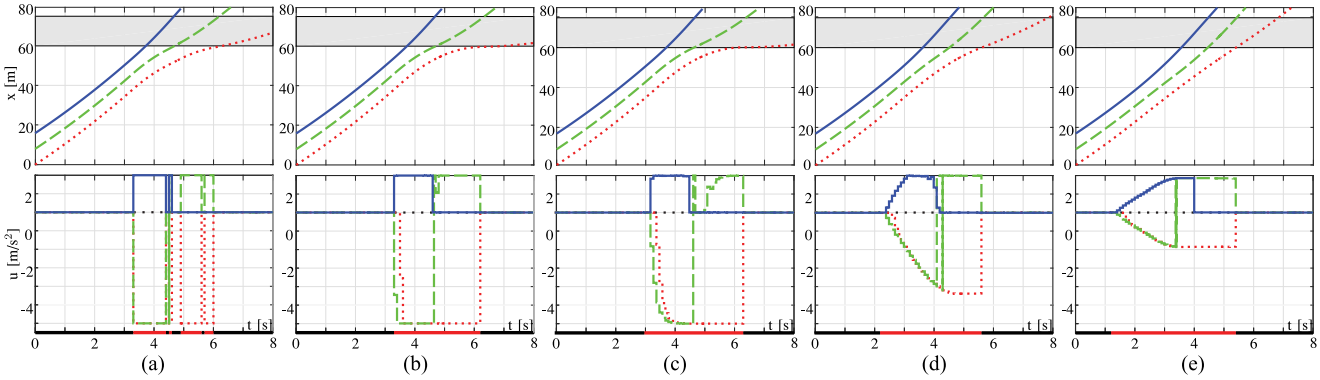

Fig. 10. Simulated three-vehicle scenarios with $\tau = 0.1$ s: (a) Non-optimized solution; the proposed optimal supervisor (Algorithm 4) with: (b) $\theta = 0.1$ s; (c) $\theta = 0.2$ s; (d) $\theta = 1$ s; (e) $\theta = 2$ s. In the bottom panels, the dotted curves represent the measured input, while the continuous curves correspond to the input given by the supervisor. The time axis is red if the supervisor is overriding the desired input, black otherwise.

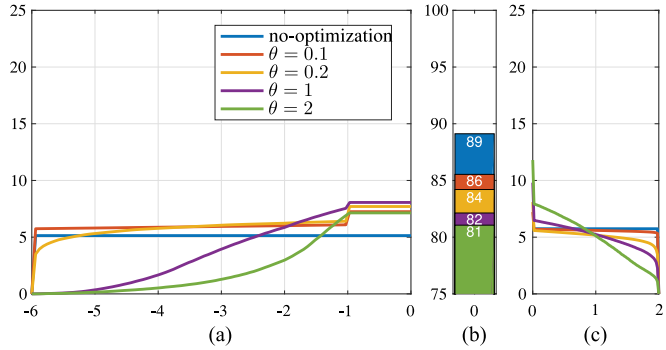

Fig. 11. Probability of having $(u(k\tau) - u_{\mathrm{des}}(k\tau))$: (a) less than; (b) equal to or; (c) greater than the value displayed on the abscissa axis. The probabilities are computed via a Monte Carlo simulation on a set of 10 000 randomized initial conditions and for different values of θ.

questing a $\mathbf{u}_{\mathrm{des}} = \mathbf{u}_{\mathrm{hyp}} = 1$, with a randomly chosen initial position $x_i(0) \in [0, 60]$ m and speed $\dot{x}_i(0) \in [v_{\min}, v_{\max}], \forall i$. The intersection corresponds to the interval $[60, 75]$ m along the vehicles' paths, the simulation time is 8 s and $\tau = 0.1$ s. This means that there are (nr. of initial conditions $\times$ nr. of vehicles $\times$ simulation time$/\tau$) samples. The results of randomized simulations are presented in Fig. 11. We consider the multiobjective optimization algorithm (15) for different values of θ, and we compare our optimal approach with the nonoptimized one. In panel (a), we show in the horizontal axis the input difference ξ and in the vertical axis the percentage of samples for which $(u_i(k\tau) - u_{i,\mathrm{meas}}(k\tau)) < \xi$, i.e., the percentage of samples for which the supervisor overrides the drivers' request with an input lower that $(u_{i,\mathrm{meas}}(k\tau) + \xi)$ with $\xi < 0$. Ideally, the curves should be as low as possible on the left-hand side, i.e., the percentage of samples for which the input mismatch is large should be as low as possible. One can see that having $\theta = 1$ s and $\theta = 2$ s greatly reduces the number of large interventions when compared to the nonoptimized solution (blue line).[1] One can also observe that the optimal solution with $\theta = 0.1$ s and $\theta = 0.2$ s does not increase the optimality level. Nevertheless, as shown in Fig. 7, the proposed solution is able to identify vehicles which need to be overridden from those who do not. In Fig. 10(c), we show the symmetric case, i.e., on the vertical axis is shown the percentage of samples for which $(u_i(k\tau) - u_{i,\mathrm{meas}}(k\tau)) > \xi$ with $\xi > 0$. As expected, increasing the value of θ reduces the number of interventions with a large mismatch with respect to the drivers' intent. Finally, we can observe in Fig. 10(b) the percentage of nonoverridden samples: increasing θ slightly increases the percentage of overrides, from 11% for the nonoptimized solution (i.e., 89% of nonoverrides) to 17% for our optimal solution with $\theta = 1$ s. However, as seen on the right- and left-hand side of Fig. 10(a) and (c), respectively, the number of very small interventions is visibly higher for the optimal solution with respect to the nonoptimized one. Increasing the value of θ increases the number of total overrides but reduces the difference $(\mathbf{u} - \mathbf{u}_{\mathrm{meas}})$. Hence, by tuning the value of θ, one can better approximate the driver's desired input and provide a more user-friendly experience. The worst computation time over the 2.430.000 samples is 0.087 s.

D. Approximate Algorithm

We present in this section simulation results for the approximate supervisor discussed in Section VII-A. We consider an eight-vehicle scenario, where initial conditions for each vehicle i is given as $x_i = (i * 8, 10), \forall i = \{0, \ldots, 7\}$. The maximum time to run the optimal algorithm for an eight-vehicle problem being 0.15 s, we have therefore defined the time stepping as $\tau = 0.2$ s. As before, it is assumed that $\mathbf{u}_{\mathrm{hyp}} = 1$ (horizontal dotted line), and that the intersection corresponds to the interval $[60, 75]$m along all vehicles' path.

Fig. 12(a) and (b) consider $\theta = 0.4$ s and $\theta = 2$ s, respectively. By optimizing the trajectories over a longer prediction horizon, one can see that the supervisor approximates better the drivers' desired inputs. Note that, as discussed before, interventions are triggered earlier as the value of θ increases. Indeed, while in

[1]The evolution of the system depends on the supervisor override as well as on the future requested input. Hence, a control strategy which is suboptimal for a given set of initial conditions may, in the long run, allow a more "gentle" override signal. Had we just compared the maximum of the infinity norm difference between the desired and override input, all strategies would have looked the same.

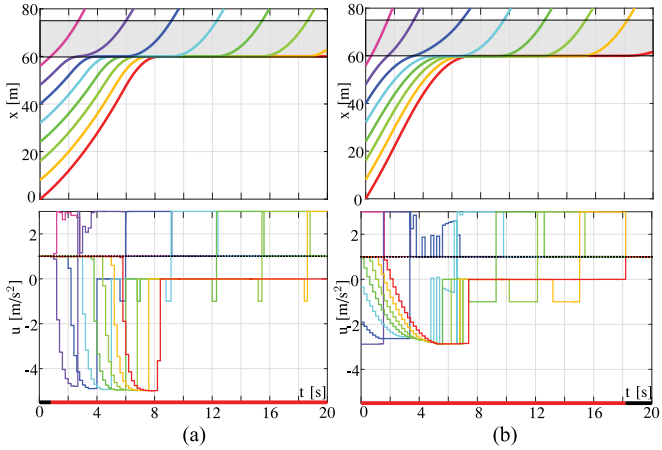

Fig. 12. Approximate supervisor Algorithm 4 applied to a simulated eight-vehicle scenario with $\tau = 0.2$ s and : (a) $\theta = 0.4$ s; (b) $\theta = 2$ s. In the bottom panels, the dotted curves represent the measured input, while the continuous curves correspond to the input given by the supervisor. The time axis is red if the supervisor is overriding the desired input, black otherwise.

Fig. 12(a) the first intervention happens at $t = 1$ s in the simulation, in Fig. 12(b) interventions are triggered immediately at the initial time. Moreover, while in Fig. 12(a) the red vehicle is not able to cross the intersection within 20 s of simulation, in Fig. 12(b) all vehicles clear the intersection.

Recall that the necessity of approximate solutions relies on the fact that the exact supervisor algorithm may be untractable for relatively small scale scenarios. However, by using the approximate algorithm proposed in [32], we are able to solve more complex problems. Here, the maximum computational time is only 2.5 times higher than the exact solution for a three-vehicle case, but for an almost three times bigger system.

IX. Discussion and Conclusions

We presented an optimal supervisor for collision avoidance at intersections leveraging results on scheduling theory. As in an optimal constrained control framework [and model predictive control (MPC) in particular], there are two main underlying aspects: 1) input/state constrained predictions, 2) a receding horizon implementation. A brief comparison is given in the following.

1) Here, the variable θ in (12) and (14) can be seen as a prediction horizon, defining how far ahead conflicts are detected. Hence, by keeping τ unchanged, one can improve its performance by increasing the value of θ. This leads to more driver-friendly, less aggressive manoeuvres. A great advantage with respect to MPC is that the complexity of the verification problem remains unchanged, independently of θ.
2) Normally, input, state, and safety constraints are formulated in optimal control-based approaches as inequalities or box conditions for all prediction instants. Hence, the number of decisions variables drastically increases for large prediction horizons and infinite horizon problems cannot easily be treated in practice. In this paper, we use a different approach and formalism: input, state, and safety constraints are incorporated in all problems through the condition $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS. By leveraging this formulation and the properties of the MCIS set, we are in fact solving an infinite horizon optimization problem that guarantees perpetual safety.
3) The supervisor routine is implemented with a stepping τ. Hence, the supervisor's output control signal is only applied for the interval $[0, \tau]$, i.e., until the next supervisor step. By regularly computing a new control policy, one can more easily cope with limited sensing/communication disturbances, as well as mitigate and compensate potential estimation errors on $\mathbf{u}_{\mathrm{hyp}}$.

Optimal conflict resolution approaches are still rare in literature. The major contribution of this paper is therefore the inclusion of optimality arguments (with respect to the drivers' desired input) into the design of our supervisor. This greatly differs from previous works in this domain such as [15]–[18], which ignore optimality and do not attempt to approximate the drivers' intent whenever the drivers' input is overridden. We also present a more generic, robust theoretical framework when compared with our previous work [25]. More precisely, we propose a solution robust to input uncertainties, easily extendable to also cope with modeling and measurement uncertainties by exploring a solution identical to [15]. In [33], a two-step optimization procedure was presented, but lacks a proof of optimality of the result of the two steps combined. Here, we prove the optimality of our solution. Finally, we also adapted the approximation strategies proposed in [18] to work as part of the optimal control design step. We are therefore able to cope with a set of eight or more vehicles in real time. Future research should approach scenarios with multiple vehicles per path and driver intent estimation techniques.

References

[1] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 84–95, Nov. 2009.
[2] B. Chen and H. H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 485–497, Jun. 2010.
[3] "Fatality analysis reporting system (FARS)," National Traffic Highway Safety Association, Washington, DC, USA, 2008.
[4] J. Broughton, P. Thomas, A. Kirk, and L. Brown, "Traffic safety basic facts 2012: Junctions," European Road Safety Observatory, London, U.K., Tech. Rep. EC FP7 project DaCoTA, 2013.
[5] A. Molinero Martinez *et al.*, "Accident causation and pre-accidental driving situations—Part 3: Summary report," Loughborough University, Loughborough, U.K., Tech. Rep. FP6-2004-IST-4 027763, 2008.
[6] A. Bender, G. Agamennoni, J. R. Ward, S. Worrall, and E. M. Nebot, "An unsupervised approach for inferring driver behavior from naturalistic driving data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3325–3336, Dec. 2015.
[7] A. Doshi and M. M. Trivedi, "Tactical driver behavior prediction and intent inference: A review," in *Proc. IEEE Conf. Intell. Transp. Syst.*, 2011, pp. 1892–1897.
[8] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer, and C. Stiller, "Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles," *IEEE Intell. Trans. Syst. Mag.*, vol. 5, no. 2, pp. 10–21, Summer 2013.
[9] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[10] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multi-agent hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.
[11] F. Fadaie and M. E. Broucke, "On the least restrictive control for collision avoidance of two unicycles," *Int. J. Robust Nonlinear Control*, vol. 16, pp. 553–574, 2006.
[12] R. Verma and D. Del Vecchio, "Semi-autonomous multi-vehicle safety: A hybrid control approach," *IEEE Robot. Autom. Mag.*, vol. 18, pp. 44–54, no. 3, Sep. 2011.
[13] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proc. ACM Conf. Hybrid Syst., Comput. Control*, 2012, pp. 145–154.
[14] M. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1162–1175, Sep. 2013.
[15] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *Proc. IEEE Conf. Decision Control*, 2013, pp. 3944–3950.
[16] H. Ahn, A. Colombo, and D. Del Vecchio, "Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles," in *Proc. Amer. Control Conf.*, 2014, pp. 867–873.
[17] A. Colombo, "A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections," in *Proc. Int. Symp. Wireless Commun. Syst.*, 2014, pp. 449–453.
[18] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, vol. 60, no. 6, pp. 1515–1527, Jun. 2015.
[19] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proc. 3rd Conf. Auton. Agents Multiagent Syst.*, 2004, pp. 530–537.
[20] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," , SAE Technical Paper, Warrendale, PA, USA, Tech. Rep. 2012-01-0292, 2012.
[21] K.-D. Kim and P. Kumar, "An MPC-based approach to provable system-wide safety and liveness of autonomous ground traffic," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3341–3356, Dec. 2014.
[22] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *Proc. Amer. Control Conf.*, 2015, pp. 763–768.
[23] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *Proc. 53rd IEEE Conf. Decision Control*, 2014, pp. 1417–1423.
[24] G. R. Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 1, pp. 8–21, Spring 2017.
[25] G. R. Campos, F. Della Rossa, and A. Colombo, "Optimal and least restrictive supervisory control: Safety verification methods for human-driven vehicles at traffic intersections," in *Proc. IEEE Annu. Conf. Decision Control*, 2015, pp. 1707–1712.
[26] D. Angeli and E. D. Sontag, "Monotone control systems," *IEEE Trans. Autom. Control*, vol. 48, no. 10, pp. 1684–1698, Oct. 2003.
[27] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.
[28] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, pp. 349–370, 1999.
[29] M. Liebner, C. Ruhhammer, F. Klanner, and C. Stiller, "Generic driver intent inference based on parametric models," in *Proc. 16th IEEE Conf. Intell. Transp. Syst.*, 2013, pp. 268–275.
[30] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer, 2008.
[31] S. A. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. Autom. Control*, vol. 55, no. 7, pp. 1646–1651, Jul. 2010.
[32] M. R. Garey, D. S. Johnson, B. B. Simons, and R. E. Tarjan, "Scheduling unit-time tasks with arbitrary release times and deadlines," *SIAM J. Comput.*, vol. 6, p. 416–426, 1981.
[33] H. Ahn, "Semi-autonomous control of multiple heterogeneous vehicles for intersection collision avoidance," M.S. thesis, Dept. Mech. Eng., Massachusetts Institute of Technology, 2014.

Gabriel Rodrigues de Campos received the Ph.D. degree in automatic control from Grenoble University/Grenoble INP, Grenoble, France, in 2012.

He is currently a Researcher with Zenuity, Göteborg, Sweden. Prior to joining the Zenuity, he was a Postdoc with the Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, and the Politecnico di Milano, Milan, Italy. His research interests include cooperative and distributed control, multiagent systems, and intelligent transportation systems.

Fabio Della Rossa received the Master's degree in mathematical engineering and the Ph.D. degree in computer science and control engineering from Politecnico di Milano, Milan, Italy, working on Bifurcation methods of complex systems, in 2008 and 2011, respectively.

He spent one year with the Department of Mathematics, Utrecht University, Utrecht, The Netherlands, working on numerical continuation methods. He is author of more than 20 papers and the book "Modeling Love Dynamics, World Scientific," 2015. He is currently an Assistant Professor with Politecnico di Milano.

Alessandro Colombo received the Master degree in engineering (Diplome d'Ingenieur) from ENSTA, Paris, France, in 2005, and the Ph.D. degree in information technology from Politecnico di Milano, Milan, Italy, in 2009.

He was a Postdoctoral Associate with the Massachusetts Institute of Technology, in 2010–2012, and he is currently an Assistant Professor with the Department of Electronics, Information and Bioengineering, Politecnico di Milano. His research interests include the analysis and control of discontinuous and hybrid systems with applications, among others, to transportation networks.