Optimal and least restrictive supervisory control: safety verification methods for human-driven vehicles at traffic intersections

Gabriel Rodrigues de Campos, Fabio Della Rossa, Alessandro Colombo

***Abstract*— We consider a cooperative conflict resolution problem at traffic intersections. Our goal is to design a least restrictive supervisor able to identify the optimal corrections to a human-decided input with respect to a given performance index, while keeping the system safe. Here, safety is formulated in terms of a maximal safe controlled invariant set. Leveraging results from scheduling theory, we characterize the preorder of the optimal solution set and propose an efficient optimization algorithm providing Pareto optimal solutions. We illustrate the application of the proposed algorithm through simulations in which vehicles crossing an intersection are optimally overridden by the supervisor only when necessary to maintain safety.**

I. INTRODUCTION

Transportation systems are increasingly relying on communication technologies and automated control in order to enable safer, smarter, and greener solutions [1]. Recent research has been focusing, among others, on the prevention and mitigation of accidents, reduction of greenhouse gas emissions, and efficiency in terms of energy and infrastructure utilization.

A particular area of interest is collision avoidance at traffic intersections [2]. Motivated by increasing levels of autonomy in road vehicles, much research effort has been focussed on cooperative and non-cooperative conflict resolution for fully and semi-automated vehicles. Several authors have recently used rule-based approaches and exploit the multi-agent systems paradigm as, for instance, in [3]–[9]. Also, [10]–[14] proposed coordination strategies based on Model Predictive Control (MPC). In particular, [10] exploits the structure of a centralized, finite-time optimal problem to propose an approximate solution, while [11], [12] considered a fully decentralized approach based on sub-optimal decision-making heuristics.

All the above references assume that a controller is either fully in charge of the vehicles, or it can set conservative bounds within which the drivers' decisions are constrained. In this paper we take a different approach. We assume that humans are in charge of driving each car, and we aim to design a supervisor which must let the drivers choose any control action, as long as this does not lead to a collision. When (and only if) this is not the case, the supervisor must correct the human decision, approximating it as best as possible while keeping the system away from conflict states. Note, however, that this solution could be coupled, in a multi-layer control structure, with existing algorithms for autonomous vehicles, by ensuring safety verification of trajectories generated by a higher level decision system.

The problem of least restrictive supervision for collision avoidance is discussed, among others, in [15]–[23], and is typically set in a framework of verification of safety specifications. Though standard general purpose algorithms exist, they are limited by 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 [18] using Scheduling Theory, and extended to more complex scenarios in [20]–[23]. Note that these papers ignore in their design optimality arguments, in the sense that, when the drivers' input is overridden, no attempt is done to approximate the drivers' intent. Clearly, this can lead to unnecessarily aggressive decelerations/accelerations.

This paper addresses this issue by providing optimal, least restrictive supervisory control for a group of human-driven vehicles. Our approach is based on the solution of two separate problems: (i) the *Verification Problem*, determining if there exists an input signal that leads all agents safely through the intersection; and (ii) the *Supervisor Problem*, returning a safe and optimal approximation of the drivers' intent if the desired input violates safety conditions. By formulating the Supervisor Problem as a multi-objective optimization problem, we identify a preorder of the set of optimal solutions, and show how to iteratively perform scheduling optimization by solving the Verification Problem. To determine the least restrictive set of control actions, we exploit the notion of maximal controlled invariant set (MCIS) and the least restrictive feedback map that keeps the system inside this set. Note that determining membership in the MCIS was proved to be NP-hard in the case of some collision avoidance problems in [18], [24]. Nevertheless, approximate solutions exist guaranteeing bounded errors with respect to the exact one.

The paper is organized as follows. Section II describes the dynamic model; Section III provides the problem formulation; in Sections IV and V we formulate the single and multi objective analog of the supervisor problem, respectively; finally, we discuss a way to trade overall performance of the supervisor with restrictiveness in Section VI. All results are illustrated by simulations in Section VII.

II. MODEL AND NOTATION

Consider the system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),\ \mathbf{y} = \mathbf{h}(\mathbf{x})$, where $\mathbf{x} \in X \subseteq \mathbb{R}^{rn}$ is the state of n agents moving on n different paths (such as in Fig. 1), with a r-order dynamics, $\mathbf{y} \in \mathbb{R}^n$ is the vector of the positions of the agents along their paths and $\mathbf{u}$ is a control input. The previous system is given by the parallel composition of n different systems which describe the longitudinal dynamics of each agent, given by $\dot{x}_i = f_i(x_i, u_i),\ y_i = h_i(x_i)$. We assume that the individual systems are monotone [25], with $\mathbb{R}_+$ (the nonnegative real line) as the positivity cone of y_i, and that the full system has unique solutions. Throughout the text, the symbols 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 space 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

This work is supported by the grant "AD14VARI02 - Progetto ERC BETTER CARS - Sottomisura B". The authors are with the DEIB at Politecnico di Milano, Italy. Email: {gabriel.rodriguesdecampos, fabio.dellarossa, alessandro.colombo}@polimi.it

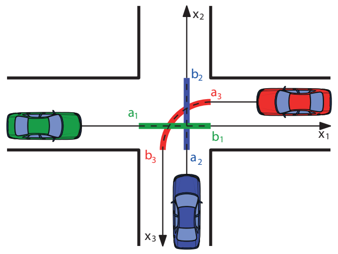

Fig. 1. Illustration of the considered scenario. Several human-driven vehicles approach an intersection following pre-defined paths.

assume that $\dot{y}_i$ is bounded to the nonnegative interval $[0, \dot{y}_{i,max}]$ for all i and that $\lim_{t\to\infty} y_i(t, u_{i,max}) = \dot{y}_{i,max}$.

III. Problem Statement

We assign to each agent an open interval (a_i, b_i), which represents the span of the intersection along the agent's path, see Fig. 1. Note that this interval must account for the physical size of the agent and of the intersection. A collision occurs when two agents verify the conditions $y_i(t) \in (a_i, b_i)$ and $y_j(t) \in (a_j, b_j)$ at the same instant t. We call *Bad Set* the subset $B \subset \mathbb{R}^n$ of collision points, defined as: $B := \{\mathbf{y} \in \mathbb{R}^n : y_i \in (a_i, b_i) \wedge y_j \in (a_j, b_j), \text{for some } i \neq j\}$. Our approach is based in two main problems: the Verification Problem and the Supervisor Problem. Their formal definition is given as follows.

Problem 1 (Verification Problem (VP))**.** Given the initial conditions $\mathbf{x_0}$ determine if there exists an input signal $\mathbf{u}$ which guarantees that $\mathbf{y}(t, \mathbf{u}, \mathbf{x}_0) \notin B$ for all $t \geq 0$.

A computationally efficient solution to VP was proposed in [18], [23] through a reduction to a job scheduling problem. The exact solution is, in general, NP-hard [18], but approximations with guaranteed error bound are available, which allow to solve the Verification Problem for very large systems.

The subset of X of all initial conditions which satisfy the Verification Problem is known in the literature as the Maximal Controlled Invariant Set (MCIS) [26]. As long as the system's state remains in the MCIS, there exists an input which avoids all collisions. Therefore, the role of the supervisor is to ensure that the state never leaves the MCIS, and a least restrictive supervisor should do so by modifying the input selected by the user (in our case, by the drivers) as little as possible and only if strictly necessary. To formally express this we use the *tangent cone* at $\mathbf{x}$ to the MCIS, denoted $\mathcal{T}_\mathbf{x}$ MCIS, which is the set of all vectors $\mathbf{v}$ such that

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

Calling $\mathbf{u}_{des} : \mathbb{R}_+ \to \mathbb{R}^n$ the drivers' intent and $\mathbf{u} : \mathbb{R}_+ \to \mathbb{R}^n$ the input returned by the supervisor, we can formalise the Supervisor Problem as follows.

Problem 2 (Supervisor Problem)**.** Given the current state $\mathbf{x}_0$ and the input signal $\mathbf{u}_{des}$, return $\mathbf{u}$ such that $f(\mathbf{x}_0, \mathbf{u}) \in \mathcal{T}_\mathbf{x}$ MCIS and such as to minimize a cost function $J(\mathbf{u}_{des}, \mathbf{u})$.

In practice, without further assumptions on the drivers' intent, at any given time the supervisor can only know a desired input *vector* (e.g., the instantaneous reading of brake and acceleration for all drivers). Therefore, we assume in the sequel that $\mathbf{u}_{des}$ is a constant signal, with value equal to the vector of the last measured desired input. This amounts, in practice, to optimize the control signal towards the best admissible reading of the driver's input during each control time interval. The above supervisor can be implemented as a discrete-time algorithm, by fixing a small positive time stepping τ and computing the input signal $\mathbf{u}$ allowed for the interval $[t, t+\tau]$ as the solution of the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u}\in\mathcal{U}} \quad & J(\mathbf{u}_{des}, \mathbf{u}) \\ \text{subject to} \quad & \mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}. \end{aligned} \tag{1}$$

Note that the constraint of the above problem is evaluated as the solution of the Verification Problem, and can therefore be addressed using the techniques discussed in [18], [23]. The standard, non-optimal implementation of the Supervisor Problem proposed in [18], [20]–[23] is equivalent to solving (1) with cost function

$$J(\mathbf{u}_{des}, \mathbf{u}) := \begin{cases} 0 \text{ if } \mathbf{u}_{des} = \mathbf{u}, \\ 1 \text{ if } \mathbf{u}_{des} \neq \mathbf{u}. \end{cases} \tag{2}$$

This corresponds to returning $\mathbf{u}_{des}$ whenever this maintains the state within the MCIS, and returning an arbitrary input $\mathbf{u}$ such that $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS otherwise: existing results do not optimize the input when $\mathbf{u} \neq \mathbf{u}_{des}$. In order to address this problem, we propose a way to select an optimal value for $\mathbf{u}$ when $\mathbf{u} \neq \mathbf{u}_{des}$, by taking as cost function the infinity norm of the deviation between the supervisor output and the desired input:

$$J(\mathbf{u}_{des}, \mathbf{u}) := \|\mathbf{u} - \mathbf{u_{des}}\|_\infty. \tag{3}$$

In the sequel, we provide a numerical strategy to optimize the above function using existing results on the Verification Problem. We then see that the input which minimizes (3) is in general not unique and, by formulating a multi-objective analog of the Supervisor Problem, we show that the set of optimal solutions has a preorder, which allows to select a optimal solution.

IV. Single objective control design

The solution to problem (1) given the cost function (3) is trivial if $\mathbf{x}(\tau, \mathbf{u}_{des}, \mathbf{x}_0) \in$ MCIS. In this case, $\mathbf{u} = \mathbf{u}_{des}$ satisfies all the problem's constraints with $J(\mathbf{u}_{des}, \mathbf{u}_{des}) = 0$. In this section, we solve problem (1) in the case when $\mathbf{x}(\tau, \mathbf{u}_{des}, \mathbf{x}_0) \notin$ MCIS.

A. Problem Reformulation

We can rewrite problem (1) with cost function (3) as follows:

$$\begin{aligned} \min_{\mathbf{u}\in\mathcal{U},\, u_{bound}\in\mathbb{R}_+} \quad & u_{bound} \\ \text{subject to} \quad & \|\mathbf{u} - \mathbf{u}_{des}\|_\infty \leq u_{bound} \\ & \mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in \text{MCIS}. \end{aligned} \tag{4}$$

Here, u_{bound} is an upper bound to (3), and minimizing it while satisfying safety constraints yields an optimal solution to (1). Let now MCIS(u_{bound}) denote the set of all states $\mathbf{x} \in X$ which satisfy the Verification Problem under the constraint

$$\|\mathbf{u}(t) - \mathbf{u}_{des}(t)\|_\infty \leq u_{bound} \text{ for } t \in [0, \tau]. \tag{5}$$

and consider the optimization problem

$$\begin{aligned} \min_{u_{bound}\in\mathbb{R}_+} \quad & u_{bound} \\ \text{subject to} \quad & \mathbf{x}_0 \in \text{MCIS}(u_{bound}). \end{aligned} \tag{6}$$

The following result holds.

Lemma 1. The optimal cost of (4) is equal to the optimal cost of (6).

Proof. Let $u^{'*}_{bound}$ and u^*_{bound} represent the optimal costs of (4) and (6), respectively. The following two arguments are true:

- $u^*_{bound} \geq u^{'*}_{bound}$: If $\mathbf{x}_0 \in$ MCIS(u^*_{bound}), then there exists $\bar{\mathbf{u}}$ such that $\|\bar{\mathbf{u}} - \mathbf{u}_{des}\|_\infty \leq u^*_{bound}$ for all $t \in [0, \tau]$ and $\mathbf{x}(\tau, \bar{\mathbf{u}}, \mathbf{x}_0) \in$ MCIS. Take now $\mathbf{u} = \bar{\mathbf{u}}$ for $t \in [0, \tau]$ and $\mathbf{u} = \mathbf{u}_{des}$ for $t > \tau$. This gives $\|\mathbf{u} - \mathbf{u}_{des}\|_\infty \leq$

u^*_{bound} and $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS. Thus, $(\bar{\mathbf{u}}, u^*_{bound})$ is a feasible solution for (4).

- $u'^*_{bound} \geq u^*_{bound}$: If $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS and $\|\mathbf{u} - \mathbf{u}_{des}\|_\infty \leq u^*_{bound}$, then $\mathbf{x}_0 \in$ MCIS(u'^*_{bound}). Therefore, u'^*_{bound} is a feasible solution for (6).

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

The advantage or rewriting (4) as (6) is that the search space of the latter is the non-negative real line, rather than a functional space. The optimal solution u^*_{bound} to (6) can be numerically computed using a bisection method (see Algorithm 1), and a full optimal solution of (4) can be retrieved by selecting an input $\mathbf{u}$ satisfying the constraints of (4) for $u_{bound} = u^*_{bound}$ (ways to construct such an input are explained, e.g., in [18]). Algorithm 1 inherits the complexity of the verification step $\mathbf{x}_0 \in \mathrm{MCIS}(u_{bound})$, since the bisection cycle is $O(1)$. Therefore, the complexity of optimally solving (1) is comparable to that of solving the Verification Problem, and any approximate polynomial-time solution of the Verification Problem directly improves the solution of (1).

Algorithm 1 Numerical solution of (6)

```
1: Initialise $U = \max_i(u_{i,max} - u_{i,min})$, $L = 0$
2: while $U - L >$ threshold do
3:     $u_{bound} = (U + L)/2$
4:     if $\mathbf{x}_0 \in \mathrm{MCIS}(u_{bound})$ then
5:         $U = u_{bound}$
6:     else
7:         $L = u_{bound}$
```

V. Multi objective control design

The optimal cost u^*_{bound} of (4) is the smallest value of the cost function (3) for which all agents can avoid collisions. There can clearly be multiple optimal solutions $\mathbf{u}$ with the same cost u^*_{bound} and, in particular, for some of these solution the cost $J_i(u_{des,i}, u_i)$ (defined as (3) restricted to agent i) for some agent may be smaller than for other solutions. In other words, there is a set of optimal solutions, and the single-agent cost functions $J_i(u_{des,i}, u_i)$ induce a preorder on this set. This solution structure is more appropriately analysed in terms of Pareto optimality in a multi-objective problem. Let us rewrite Problem (1) as the following multi-objective optimization problem:

$$\begin{aligned} \min_{u_i \in \mathcal{U}_i} \quad & J_i(u_{des,i}, u_i), \forall\, i \\ \text{subject to} \quad & \mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in \mathrm{MCIS}, \end{aligned} \tag{7}$$

Definition 1. An admissible solution $\mathbf{u}$ of (7) is called *weak Pareto optimal* if there exists no admissible solution $\mathbf{u}'$ such that $J_i(u_{des,i}, u'_i) < J_i(u_{des,i}, u_i)$ for all i; it is called *Pareto optimal* if there exists no admissible solution $\mathbf{u}' \neq \mathbf{u}$ such that $J_i(u_{des,i}, u'_i) \leq J_i(u_{des,i}, u_i)$ for all i and $J_i(u_{des,i}, u'_i) < J_i(u_{des,i}, u_i)$ for at least one i (see Fig.2).

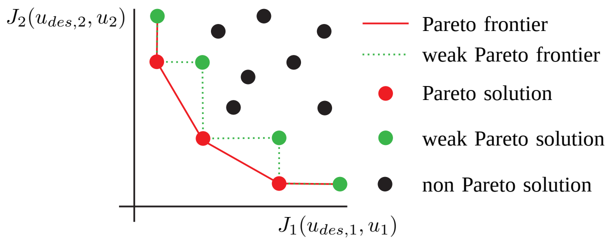

Fig. 2. Illustration of Pareto and weak-Pareto solutions, accordingly to Def.1.

Pareto optimal solutions are by definition not comparable in the preorder induced by (7). Therefore, in the absence of further hypotheses, all Pareto optimal solutions are equally good. The following result holds.

Lemma 2. All solutions of (1) are weak Pareto Optimal for (7).

Proof. The proof is by contradiction. Assume that there is an optimal solution $\mathbf{u}$ of (1) that is not Pareto optimal for (7). This means that exists a solution $\mathbf{u}'$ of (7) such that $J_i(u_{des,i}, u'_i) < J_i(u_{des,i}, u_i)$ for all i. Then $J(\mathbf{u}_{des}, \mathbf{u}') < J(\mathbf{u}_{des}, \mathbf{u})$ in (1), contradicting the optimality of $\mathbf{u}$. $\square$

By the above lemma, any solution of (1) is at least weak Pareto optimal. It is however interesting to select, among optimal solutions, one which is Pareto optimal, see Fig.2. In the rest of this section we discuss a way to identify one such solution. Our approach exploits a representation of the constraint of (7) in terms of a scheduling problem, following the idea introduced in [18]. We briefly introduce the scheduling equivalence in Section V-A, then we discuss the optimization algorithm.

A. Verification problem vs. Scheduling problem

Define for each agent i such that $y_i(0) \leq a_i$ the quantities $R_i := \inf_{u_i \in \mathcal{U}_i}\{t : y_i(t, u_i) = a_i\}$, $D_i := \sup_{u_i \in \mathcal{U}_i}\{t : y_i(t, u_i) = a_i\}$, and set $R_i = D_i = 0$ if $y_i(0) > a_i$. These two quantities are, respectively, the minimum and maximum time at which the output of system i can reach 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 such that $y_i(0) \leq a_i$, given a real number T_i, define $P_i(T_i) := \inf_{u_i \in \mathcal{U}_i}\{t : y_i(t, u_i) = b_i\}$, with constraint $y_i(t, u_i) \leq a_i\ \forall\, t < T_i$.

If the constraint cannot be satisfied, set $P_i(T_i) := \infty$. If $y_i(0) \in (a_i, b_i)$ define $P_i(T_i) := \inf\{t : y_i(t, u_{i,max}) = b_i\}$, and if $y_i(0) \geq b_i$ define $P_i(T_i) := 0$. $P_i(T_i)$ is the earliest time that i can reach b_i, if it does not pass a_i before T_i. A scheduling problem consists in assigning jobs to a resource satisfying given requirements. Using the above quantities, we can write the Verification Problem 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. *Given an initial condition $\mathbf{x}_0$, $\mathbf{x}_0 \in$ 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:*

$$R_i \leq T_i \leq D_i, \tag{8}$$

and for all (i, j), if $x_i(0) < b_i$, then

$$T_i \geq T_j \Rightarrow T_i \geq P_j(\mathbf{T}). \tag{9}$$

The reader can refer to [18] for the proof in the case where $\dot{y}$ is bounded above 0. An extension of the proof to $\dot{y} \in [0, \dot{y}_{max}]$ is simple. Notice that, in the above definitions, the quantities R_i, D_i and P_i are all dependent on the set $\mathcal{U}_i$. In the presence of constraint (5), then, these quantities become a function of the constraining quantity u_{bound}. The following definition is introduced.

Definition 2 (Scheduling Problems)**.** Letting SP denote a scheduling problem defined by (8) and (9), we write

- SP(u_{bound}) when the scheduling quantities are computed under the constraint (5).
- SP$(u_{1,bound}, \ldots, u_{n,bound})$ when the constraint (5) is different for different agents.

- $\mathbf{T} \in \text{SP}(u_{bound})$ if $\mathbf{T}$ is a feasible schedule of SP(u_{bound}).
- SP(L, u_{bound}) when a restriction of SP(u_{bound}) to a subset L of the agents $\{1, \ldots, n\}$ is considered.

In this notation and by Theorem 1, the constraint of problem (6) can be written as $\exists \mathbf{T} : \mathbf{T} \in \text{SP}(u_{bound})$. We can now prove a simple property regarding the dependence of SP on u_{bound}:

Lemma 3. Consider the quantities R_i, D_i and $P_i(T_i)$ of SP(u_{bound}), and R'_i, D'_i and $P'_i(T_i)$ of SP(u'_{bound}) with $u'_{bound} < u_{bound}$. 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 SP(u_{bound}) is a relaxation of SP(u'_{bound}). □

In what follows, we also consider an extension of the scheduling problem defined by (8) and (9) where jobs cannot be executed during specified time intervals, which are known as *inserted idle times* (iit) [21]. Given a set of inserted idle times IIT:= $\{[\alpha_1, \beta_1], [\alpha_2, \beta_2], \ldots\}$, this amounts to adding to the above the additional constraint

$$(T_i, P_i(T_i)) \cap (\alpha_j,\ \beta_j) = \emptyset,\ \forall\, i, j. \qquad (10)$$

This problem will be denoted SP(u_{bound}, IIT).

B. Multi-objective optimization algorithm

In this section, we discuss an algorithmic solution to (7) which identifies a Pareto optimal solution. The following definitions are used to discuss the algorithm.

Definition 3 (Tight set)**.** Consider a schedule $\mathbf{T} \in \text{SP}(u_{bound}, \text{IIT})$. We say that an ordered set of jobs and inserted idle times $i = \{1, \ldots, m\}$ is tight if the following conditions are satisfied: (i) 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 $\alpha_i = \beta_{i-1}$; (ii) if the first element is a job it starts exactly at its release time, i.e., at R_1; (iii) if the last element is a job, it starts exactly at its deadline D_m.

In other 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 $\mathbf{T} \in \text{SP}(u_{bound}, \text{IIT})$, we can identify a subset of jobs which do not satisfy constraints (8), (9), or (10) if u_{bound} is reduced, unless we change the order with which they are scheduled in $\mathbf{T}$. We call these jobs *constrained*, and formally define them as follows, explicitating with $P_i(u_{bound})$ and $D_i(u_{bound})$ the dependence of the scheduling quantities of problem $\text{SP}(u_{bound})$ on u_{bound}.

Definition 4 (Constrained and constraining jobs)**.** A tight job i is *constrained* **in** a schedule $\mathbf{T}$ **for** a problem SP(u_{bound},IIT) if **i)** it is followed by another tight job j and $P_i(T_i, u'_{bound}) > T_j$ for any $u'_{bound} < u_{bound}$, or **ii)** it is followed by an IIT $[\alpha, \beta]$ and $P_i(T_i, u'_{bound}) > \alpha$ for any $u'_{bound} < u_{bound}$, or **iii)** $T_i > D_i(u'_{bound})$ for any $u'_{bound} < u_{bound}$.
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, see Fig. 3.

We can think of the constraining jobs for a schedule $\mathbf{T} \in \text{SP}(u_{bound}, \text{IIT})$ as those jobs which limit the minimum value that u_{bound} can take while allowing $\mathbf{T}$ to be adapted to be

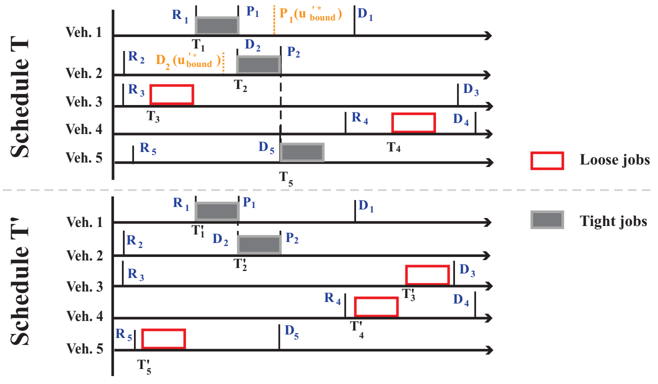

Fig. 3. Illustration of two feasible schedules for $\text{SP}(u^*_{bound}, \text{IIT})$. Jobs 1 and 2 are constrained in both, while job 5 is constraining in T and not tight in T'. Schedule T' is then the constraint-minimal between the two.

feasible in $\text{SP}(u_{bound}, \text{IIT})$ without changing the relative order of jobs and iit's.

Definition 5 (Constraint-minimal schedule)**.** Consider a schedule $\mathbf{T} \in$ SP. The schedule is constraint-minimal if no other schedule $\mathbf{T}' \neq \mathbf{T}$, $\mathbf{T}' \in$ SP has a set of constrained jobs that is a strict subset of that of $\mathbf{T}$. See Fig. 3.

Using the above definitions we can prove a useful result, based on the following construction.

- Consider a schedule $\mathbf{T} \in \text{SP}(u^*_{bound}, \text{IIT})$, where u^*_{bound} is the optimal cost of (4) with constraint $\exists\, \mathbf{T} : \mathbf{T} \in \text{SP}(u_{bound}, \text{IIT})$, and assume that $\mathbf{T}$ is constraint-minimal.
- Define a set C of constrained jobs and L of jobs that are not constrained in $\mathbf{T}$ for $\text{SP}(u^*_{bound}, \text{IIT})$, and define a new set $\text{IIT}' := \text{IIT} \cup \{[T_i, P_i(T_i)] \forall i \in C\}$.
- Call u'^*_{bound} the optimal cost of (6) with constraint $\exists\, \mathbf{T}' : \mathbf{T}' \in \text{SP}(L, u_{bound}, \text{IIT}')$.
- Finally, consider the scheduling problem $\text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$, where $u''^*_{bound,i} := u^*_{bound}$ if $i \in C$, and $u''^*_{bound,i} := u'^*_{bound}$ if $i \in L$.

Lemma 4. The set of constrained jobs in $\mathbf{T}$ for $\text{SP}(u^*_{bound}, \text{IIT})$ is a subset of the set of constrained jobs in any $\mathbf{T}'' \in \text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$ for $\text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$.

Proof. This is a consequence of selecting a constraint-minimal schedule $\mathbf{T}$. First of all, notice that (i) $u'^*_{bound} < u^*_{bound}$, by Lemma 3 and since u'^*_{bound} is computed by removing from SP all constraining jobs, and that (ii) for any $\mathbf{T}'' \in \text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$, $\mathbf{T}'' \in \text{SP}(u^*_{bound}, \text{IIT})$. The only way that a job which is constrained in $\mathbf{T}$ for $\text{SP}(u^*_{bound}, \text{IIT})$ can be not constrained in $\mathbf{T}''$ for $\text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$, 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 $\text{SP}(u^*_{bound}, \text{IIT})$. We have defined $u''^*_{bound,k} = u'^*_{bound} < u^*_{bound}$, therefore this would imply $\mathbf{T}'' \notin \text{SP}(u''^*_{bound,1}, \ldots, u''^*_{bound,n}, \text{IIT})$. □

Algorithm 2 is based on the construction above. As we have seen in Lemma 3, decreasing the value of u_{bound} tightens the constraints of SP(u_{bound}). As a consequence, we can interpret the optimal cost of (4) as the value u^*_{bound} for which a subset of jobs verify the constraints exactly, i.e, would not be schedulable for a smaller value of u_{bound} . If we remove these job from the optimization problem (but reserve their execution time as iit),

Algorithm 2 Multi-objective optimization algorithm

```
1:  Initialise $\mathbf{L} = \{1, \dots, n\}$, IIT$= \emptyset$, $V^* = \infty$
2:  set $k = 0$
3:  while $\mathbf{L}$ is nonempty or $V^* > 0$ do
4:      $k = k + 1$
5:      Optimization step: Solve problem (6) with constraint
6:                         $\exists \mathbf{T} : \mathbf{T} \in \text{SP}(L, u_{bound}, IIT)$ and
7:                         call $V^*$ its optimal cost
8:      Selection step: Select a schedule $\mathbf{T} \in \text{SP}(L, V^*, IIT)$
9:                      that is constraint-minimal
10:     Reduction step:
11:     for all jobs $i \in \mathbf{L}$ that are constrained
12:             for $\mathbf{T}_k$ in SP$(L, V^*, IIT)$ do
13:         remove $i$ from $\mathbf{L}$
14:         add the interval $[T_i, P_i(T_i)]$ to IIT
15:         set $J_i^* := V^*$, $T_i^* := T_i$
16: return $(J_1^*, \dots, J_n^*)$, $(T_1^*, \dots, T_n^*)$
```

we can solve (4) for the remaining jobs and find a lower optimal cost. Iterating this reasoning we form a solution to the multi-objective problem (7). Note that the complexity of Algorithm 2 is defined by the complexity of the optimization step, which in turn is leaded by the complexity of the test at line 4 in Algorithm 1.

Theorem 2. *Algorithm 2 provides a Pareto optimal solution to* (7).

Proof. The algorithm is implementing the process described before Lemma 4, and it returns a schedule $\mathbf{T}^*$ and a solution to (7) in terms of a set of optimal costs $J_1^*, \dots, J_n^*$. From Lemma 4, we can conclude that all jobs for the schedule $\mathbf{T}^* \in \text{SP}(J_1^*, \dots, J_n^*)$ are constrained, or have $J_i^* = 0$. In both cases, it is not possible to find a feasible schedule for a problem $\text{SP}(J_1'^*, \dots, J_n'^*)$ with $J_i'^* < J_i^*$ for at least one i, therefore $J_1^*, \dots, J_n^*$ is a Pareto optimal solution. □

VI. Positive predictive horizon

Though the proposed algorithm does not rely on a MPC-based formulation, the bridges between the two approaches are evident and worth of a discussion. Two aspects are shared between them: (i) the notion of prediction horizon; (ii) the concept of input/state constrained predictions.

As for MPC, the variable τ in (4) and (6) effectively acts as a prediction horizon defining how far ahead conflicts are detected. By verifying that $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS, we are in fact imposing a specification on $\mathbf{u}$ over the interval $[0, \tau]$.

Keeping the time-stepping of a supervisor unchanged, we can therefore improve its performance by increasing the value of τ in the constraint of the optimal output computation. This may lead to more driver-friendly, less aggressive manoeuvres. The trade-off is on the restrictiveness of the supervisor, since for large values of τ interventions may be triggered earlier than strictly necessary[1].

Another important comparison aspect is the consideration of state and input constrained predictions. Note that while in a MPC framework input, state and ultimately safety constraints are explicitly formulated as inequalities or box conditions, equivalent constraints are incorporated here into the definition of the MCIS set, and implicitly enforced in the different formulations through the condition $\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}_0) \in$ MCIS.

[1]Note that, unlike for MPC, a larger value of τ will not increase the computational complexity of the proposed protocol. Indeed, the complexity of the Verification Problem is independent of τ (which only defines the state's projection horizon) and therefore the complexity of the optimization problems remains unchanged.

VII. Simulation results

In order to show the efficiency of the proposed optimal supervisory control, we considered in the sequel a three-vehicle scenario as depicted in Fig. 1. Assume that the longitudinal dynamics of all vehicles, travelling over three different paths, are described by double integrator dynamics given by: $\ddot{x}_i(t) = u_i(t)$ and $y_i(t) = x_i(t)$, where $\dot{x}_i \in [0m/s, 17m/s]$ and $u_i \in [-4\,m/s^2, 2\,m/s^2]$, $\forall i$. In all simulations the initial conditions of the system are $\mathbf{x} = [(0, 11), (32, 12), (35, 10)]$, and the supervisors run with a time stepping of $0.1s$. To simplify the interpretation of the results, we have assumed that the drivers of all vehicles always request an input equal to 0.5 (horizontal dashed line in all figures), and that the intersection corresponds to the interval $[60, 75]m$ along all vehicles' path (grey boxes in all figures).

In Fig. 4(a), we show the behaviour of a scheduling-based supervisor such as the one presented in [18], which does not perform any optimization of the trajectory once the boundary of the capture set is reached. At $t = 1.1$ seconds into the simulation, the supervisor detects that vehicles are about to leave the MCIS and intervenes to correct their trajectory. By construction, the override input takes the extremal values $u_i = +2$ and $u_i = -4$ until a first collision is averted at $t = 3.1s$, and a second one is averted at $t = 6.2s$.

Fig. 4(b) considers the same scenario when the single-objective optimal supervisor (6) is implemented with $\tau = 0.1s$. By optimizing the intervened trajectories, the supervisor provides a slightly better approximation of the drivers' desired inputs while avoiding two consecutive 2-vehicle conflicts from $t = 1.1s$ to $t = 3.2s$ and from $t = 5.4s$ to $t = 6.5s$. Note that for both conflicts the only trajectories able to avoid collisions are obtained with extremal inputs. It is worth mentioning that this is always the case for trajectories sliding on the capture-set of a two-vehicle problem.

Fig. 5(a) shows the resulting trajectories when a multi-objective optimal supervisor (7) is implemented with $\tau = 0.1s$. As expected, the performance of the supervisor improves. More precisely, the supervisor only overrides the blue and green vehicles with respect to the first collision, allowing the red vehicle to continue its desired trajectory. The overriding of the red vehicle only occurs at $t = 3.2s$, i.e., two seconds later that in previous examples; the performance of the green vehicle is also improved (from $t = 3.2s$ to $t = 6s$). Without needing to override unnecessarily the green vehicle, the multi-objective

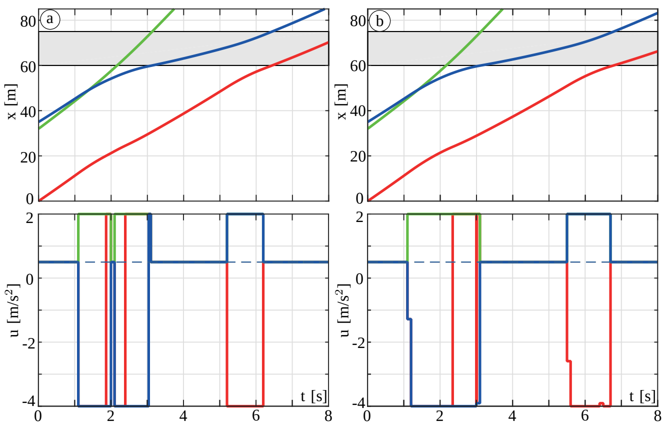

Fig. 4. (a) Traditional supervisor based on (1) and cost (2) with $\tau = 0.1s$; (b) Single-objective optimal supervisor with $\tau = 0.1s$.

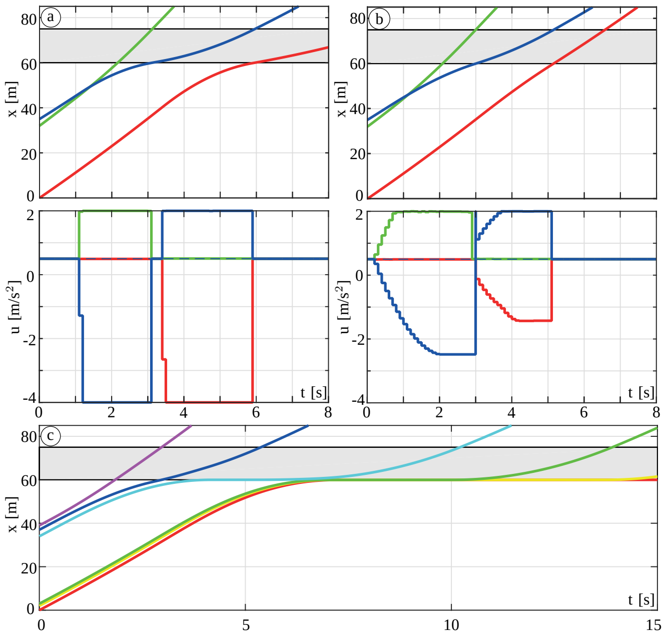

Fig. 5. Multi-objective optimal supervisor: (a)with $\tau = 0.1s$; (b) with $\tau = 1s$; (c) with an EDD scheduling with $\tau = 1s$

supervisor is, as expected, less restrictive/invasive. Globally, one can easily observe that the approximation of the drivers' desired input are significantly improved.

Fig. 5(b) shows the trajectories of a multi-objective optimal supervisor, where $\tau = 1s$. As expected, the supervisor overrides the desired input about $1s$ earlier than in the other simulations, but the resulting override is less aggressive, since the maximum deceleration reached is $u_i = -2.5$, instead of $u_i = -4$ as in previous cases.

Finally, Fig. 5(c) shows the trajectories of a six-vehicle system when a multi-objective optimal supervisor using a Earliest Due Date (EDD) scheduling algorithm. EDD consists in choosing, among all possible job orders, a schedule where the jobs with the earliest deadlines are executed first [27]. One can easily observe that while actions are taken, in the early stage of the simulation, to avoid a collision between the purple, blue and cyan vehicles, the remaining vehicles keep their desired motion profile until $t = 5s$. From this point onwards, due to the risk of a collision between the green, yellow and red vehicles, the supervisor takes control of the different vehicles until conflicts have been solved. The maximum time to run the optimization algorithm was $0.1s$ on a 2.8 GHz, 16 Gb RAM laptop running on Windows 8.

VIII. Conclusions and perspectives

In this paper, we discussed the design of optimal, least restrictive supervisors for intersection collision avoidance. We leveraged results on scheduling theory to construct two algorithms that compute the optimal corrections to the drivers' input necessary to avoid a collision, one providing an optimal solution, the other a Pareto optimal one. The complexity of these algorithms is inherited from that of solving a Verification Problem, which was discussed in [18], and can therefore take advantage of efficient solutions to the Verification Problem. The definition of the supervisor implicitly introduces a prediction-time horizon. We exploited this to further improve our result, allowing to change this horizon to tune the trade-off between overall performance and restrictiveness.

References

[1] S. Behere, M. Törngren, and D.-J. Chen, "A reference architecture for cooperative driving," *Journal of Systems Architecture*, vol. 59, no. 10, Part C, pp. 1095 – 1112, 2013.

[2] P. Alexander, D. Haley, and A. Grant, "Cooperative intelligent transport systems: 5.9-ghz field trials," *Proc. IEEE*, vol. 99, pp. 1213–1235, 2011.

[3] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in 3^{rd} *Conference on Autonomous Agents and Multiagent Systems*, 2004.

[4] ——, "A multiagent approach to autonomous intersection management." *Journal Artif. Intell. Res.(JAIR)*, vol. 31, no. 1, pp. 591–656, 2008.

[5] A. De La Fortelle, "Analysis of reservation algorithms for cooperative planning at intersections," in *IEEE Conference on Intelligent Transportation Systems*, 2010.

[6] H. Kowshik, D. Caveney, and P. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Trans. on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, 2011.

[7] J. Gregoire, S. Bonnabel, and A. De La Fortelle, "Priority-based intersection management with kinodynamic constraints," in *European Control Conference*, 2014.

[8] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Vehicular networks for collision avoidance at intersections," SAE Technical Paper, Tech. Rep., 2011.

[9] ——, "Intersection management using vehicular networks," SAE Technical Paper, Tech. Rep., 2012.

[10] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *American Control Conference*, 2015.

[11] G. R. Campos, P. Falcone, and J. Sjöberg, "Autonomous cooperative driving: a velocity-based negotiation approach for intersection crossing," in *IEEE Conference on Intell. Transportation Systems*, 2013.

[12] G. R. Campos, P. Falcone, H. Wymeersch, R. Hult, and J. Sjöberg, "A receding horizon control strategy for cooperative conflict resolution at traffic intersections," in *IEEE Conf. on Decision and Control*, 2014.

[13] K.-D. Kim, "Collision free autonomous ground traffic: A model predictive control approach," in *ACM/IEEE Conf. on Cyber-Physical Systems (ICCPS)*, 2013, pp. 51–60.

[14] K.-D. Kim and P. Kumar, "An mpc-based approach to provable systemwide safety and liveness of autonomous ground traffic," *IEEE Trans. on Automatic Control*, vol. 59, no. 12, pp. 3341–3356, 2014.

[15] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multi-agent hybrid systems," *IEEE Trans. on Automatic Control*, vol. 43, pp. 509–521, 1998.

[16] F. Fadaie and M. E. Broucke, "On the least restrictive control for collision avoidance of two unicycles," *Int. Journal of Robust Nonlinear Control*, vol. 16, pp. 553–574, 2006.

[17] R. Verma and D. Del Vecchio, "Semiautonomous multivehicle safety: A hybrid control approach," *IEEE Robot. Autom. Mag.*, vol. 18, pp. 44–54, 2011.

[18] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *ACM Conf. on Hybrid Systems: Computation and Control*, 2012.

[19] M. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. on Intell. Transportation Systems*, 2013.

[20] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *IEEE Conf. on Decision and Control*, 2013.

[21] H. Ahn, A. Colombo, and D. Del Vecchio, "Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles," in *American Control Conference*, 2014.

[22] A. Colombo, "A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections," in *International Symposium on Wireless Communication Systems*, 2014.

[23] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. on Automatic Control*, 2015.

[24] S. A. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. on Automatic Control*, vol. 55, pp. 1646–1651, 2010.

[25] D. Angeli and E. D. Sontag, "Monotone control systems," *IEEE Trans. on Automatic Control*, vol. 48, pp. 1684–1698, 2003.

[26] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, pp. 349–370, 1999.

[27] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.