Least restrictive supervisors for intersection collision avoidance: A scheduling approach

Alessandro Colombo, Member, IEEE, and Domitilla Del Vecchio, Member, IEEE

Abstract—We consider a cooperative conflict resolution problem that finds application, for example, in vehicle intersection crossing. We seek to determine minimally restrictive supervisors, which allow agents to choose all possible control actions that keep the system safe, that is, conflict free. This is achieved by determining the maximal controlled invariant set, and then by determining control actions that keep the system state inside this set. By exploiting the natural monotonicity of the agents dynamics along their paths, we translate this problem into an equivalent scheduling problem. This allows us to leverage existing results in the scheduling literature to obtain both exact and approximate solutions. The approximate algorithms have polynomial complexity and can handle large problems with guaranteed approximation bounds. We illustrate the application of the proposed algorithms through simulations in which vehicles crossing an intersection are overridden by the supervisor only when necessary to maintain safety.

Index Terms—Maximal controlled invariant set, multi-agent system, safety, verification, scheduling, supervisory control, collision avoidance, intersection.

I. INTRODUCTION

Transportation systems are increasingly relying on communication technologies and automated control in order to make driving safer and more enjoyable [1]. A particular area of focus is the design of driver assist systems that use vehicle to vehicle and/or vehicle to infrastructure communication to prevent collisions at traffic intersection [2]–[6]. Two key requirements of these systems are that they must guarantee collision-free intersection crossing (safety), while intervening as little as possible (least restrictiveness). The latter requirement is especially important in ground transportation since drivers are expected to be in control of their vehicles at all times and overrides are accepted only in case of extreme danger. At the same time, algorithms that determine whether drivers’ desired actions are safe and provide a set of possible safe controls must be computationally efficient, since they must run in real-time in the presence of a potentially large number of vehicles.

Typically, least restrictive sets of control actions are determined by calculating the maximal (safe) controlled invariant set and the least restrictive feedback map that keeps the system inside this set. The maximal controlled invariant set is the largest set of states (with respect to set inclusion) for which there exists an input that avoids conflicts at all positive times. Determining the maximal controlled invariant set is often a computationally difficult problem, and has been proved to be NP-hard in the case of some collision avoidance problems [7], [8]. A number of exact algorithms have been proposed, whose application to systems with more than a few agents is not practical [9]–[21]. Some of these results are applicable only to the two-agent conflict resolution problem [5], [16], [19], and the others have exponential complexity in the size of the state space. Hence, research has been focusing on methods to approximate the maximal controlled invariant set with more efficient algorithms [22]–[24]. In many cases, even the termination of the algorithm that computes the maximal controlled invariant set is not guaranteed, and work has been done to identify classes of systems that allow to prove termination [25], [26].

In this paper, we design least restrictive supervisors for collision avoidance at traffic intersections. Our algorithms do not return a specific control action, and are not intended to optimize a performance metric, but rather to determine the largest set of (infinite horizon) control actions that avoid any conflict. We propose a new paradigm for the design of least restrictive supervisors that leverages the distinctive structure of the application to obtain exact and approximate algorithms for collision avoidance at intersections involving an arbitrary number of agents. Specifically, we exploit the fact that agents evolve unidirectionally along their paths through an intersection (Fig. 1) and that they have (nonlinear) monotone dynamics [27]. Under these assumptions, we demonstrate that determining whether a state belongs to the maximal controlled invariant set—which we call the verification problem (VP)—is equivalent to solving a scheduling problem (SP) [28]. While VP is defined over a function space, SP requires a search over a finite set of strings; it is thus simpler to handle. We then leverage and adapt results from the scheduling literature to determine exact and approximate solutions to the verification problem, and we use the solution of the verification problem to synthesize a least restrictive supervisor. This supervisor leaves the agents freedom of choice unless such a choice will compromise safety at some future time. We provide algorithmic procedures, which are guaranteed to terminate, that solve the verification problem. Approximate solutions use algorithms with polynomial complexity and can be efficiently used for real-time control of systems involving a large number of vehicles.

A. Colombo is with DEIB at Politecnico di Milano, Italy, e-mail: alessandro.colombo@polimi.it.
D. Del Vecchio is with the ME Dept. at MIT, MA, USA, email: dlv@mit.edu.
agents. Tight approximation bounds are provided to measure the difference between the approximate solution and the exact one.

There is an extensive literature on conflict resolution [29]–[36] and on probabilistic conflict prediction [37]–[39]. The papers on conflict resolution address problems that are close to the one tackled here, but with the following significant differences. The algorithms developed in [29], [30], [32]–[36] can handle more than two agents, however they do not provide a least restrictive solution, which is instead provided by our algorithms. The algorithms in [31] are based on a mixed integer linear programming formulation, potentially suitable to compute least restrictive solutions. However, their complexity scales exponentially in the number of agents, limiting their application to systems with a small number of vehicles. Our approach instead allows to derive approximate solutions with polynomial complexity and a provable approximation bound, which can be applied to very large systems.

The paper is organised as follows. In Section II, we introduce the model and VP. In Section III, we introduce some scheduling concepts and terminology, and we provide some basic results that will be used throughout the paper. These results will set the bases to decouple the (hard) problem of deciding in which order agents can cross the intersection from the (easy) problem of choosing how they can cross it once the order is fixed. Section IV contains the main results: we introduce SP; we prove equivalence of VP and SP; and we provide an exact and an approximate solution to SP. Section V uses these solutions for the synthesis of a least restrictive supervisor. Section VI presents an application example.

II. SYSTEM DEFINITION AND PROBLEM STATEMENT

We consider conflict resolution problems such as the one depicted in Fig. 1. We model agents as point masses moving along a specified path and governed by Newton’s law:

\[
\dot{x}_i = f_i(x_i, u_i),
\]

with \( i \in \{1, \ldots, n\} \), where \( x_i \in X_i \subseteq \mathbb{R} \) is the position and \( u_i \in U_i \subseteq \mathbb{R}^p \) is the control input. We use the following notation throughout the paper. The symbols \( x_i \) and \( u_i \) are used both for signals (functions of time) and signal values (vectors in a Euclidean space). When the dependence of \( x_i \) on the input needs to be made explicit, we write \( x_i(u_i) \). The value of \( x_i \) at time \( t_0 + t \) starting from \( (x_i(t_0), \dot{x}_i(t_0)) \), with input \( u_i \) in \([t_0, t_0 + t]\), is denoted \( x_i(t, u_i, x_i(t_0), \dot{x}_i(t_0)) \). The same notation is used for \( \dot{x}_i \). When the initial state is not important we use the shorter notation \( x_i(t, u_i) \) and \( \dot{x}_i(t, u_i) \). Boldface \( x \) and \( u \) denote the vectors \((x_1, x_2, \ldots, u_1, u_2, \ldots)\), while \( u_i \) denotes the \( j \)-th component of vector \( u_i \).

We call \( \mathcal{U} \) the set of input signals \( u \), and we assume that it contains at least the piecewise smooth functions with a finite number of discontinuities. Given a signal \( u_i \), we write \( u_i \geq u_i' \) if \( u_i(t) \geq u_i'(t) \) for all \( j \) and for all \( t \geq 0 \), and \( u_i \geq u_i' \) if \( u_i(t) \geq u_i'(t) \) for all \( j \) and for all \( t \geq 0 \) and there exists at least one \( j \) such that \( u_i(t) > u_i'(t) \) for all \( t > 0 \). Similarly, given a vector \( u_i \), let \( u_i \geq u_i' \) if \( u_i(j) \geq u_i'(j) \) for all \( j \), and \( u_i \geq u_i' \) if \( u_i(j) \geq u_i'(j) \) for all \( j \) and there exists at least a \( j \) such that \( u_i(j) > u_i'(j) \). The same notation is used for \((x_i, \dot{x}_i)\).

We make the following assumptions:

\( A.1 \) \( U_i \) has a unique minimum \( u_{i,min} \) and a unique maximum \( u_{i,max} \) in the order \( \succeq \) defined above, and \( f_i(\dot{x}_i, u_i) \) is non-decreasing in \( u_i \).

\( A.2 \) system (1) has unique solutions, depending continuously on initial conditions and parameters.

\( A.3 \) \( \dot{x}_i \) is bounded to a strictly positive interval \([\hat{x}_{i,min}, \hat{x}_{i,max}]\) with nonempty interior (agents always move forward).

\( A.4 \) \( f_i(\dot{x}_i, u_i) \) is bounded for all \( x_i \in [\hat{x}_{i,min}, \hat{x}_{i,max}] \), \( u_i \in U_i \),

\[
\lim_{t \to \infty} \dot{x}_i(t, u_i, \hat{x}_{i,max}) = \hat{x}_{i,max} \quad (\text{min and max velocities are attained at least asymptotically by applying } u_{i,min} \text{ and } u_{i,max}).
\]

\( A.5 \) dynamics of agents on the same path are identical, i.e., \( f_i = f_j \) if \( [u_{i,min}, u_{i,max}] = [u_{j,min}, u_{j,max}] \), \([\hat{x}_{i,min}, \hat{x}_{i,max}] = [\hat{x}_{j,min}, \hat{x}_{j,max}]\).

Before proceeding, we briefly comment on the above assumptions. As shown in [27], (A.1) implies that

\[
(x_i(0), \dot{x}_i(0)) \succeq (x_i'(0), \dot{x}_i'(0)), \quad u_i \succeq u_i',
\]

\[
(x_i, \dot{x}_i) \succeq (x_i', \dot{x}_i'),
\]

while the following monotonicity property follows easily:

\[
(x_i(0), \dot{x}_i(0)) \succ (x_i'(0), \dot{x}_i'(0)) \quad \text{and} \quad u_i \succeq u_i' \Rightarrow x_i > x_i'.
\]

This means that, if agent \( i \)'s position and velocity are not smaller than those of agent \( j \), and agent \( i \) applies an input not smaller than that of agent \( j \), then the ordering between the two agents is preserved. Our model therefore belongs to the class of monotone systems [27]. This is a natural assumption given the physics of the problem. Assumption (A.3) requires that the dynamics of (1) make the interval \([\hat{x}_{i,min}, \hat{x}_{i,max}]\) invariant irrespective of the input (note that (1) is not required to be smooth, so saturation is allowed). Positive minimum velocity is not a restrictive assumption, given the problem at hand, since all agents that can stop before the intersection avoiding a rear end collision can be removed from the verification problem. Finally, (A.6) implies that, if agents \( i \) and \( j \) are on the same path and \((x_i(0), \dot{x}_i(0)) = (x_j(0) + d, \dot{x}_j(0))\) for some \( d \in \mathbb{R} \), then \( x_i(t, u_i) + d = x_j(t, u_j) \) for all \( t \geq 0 \) provided \( u_i = u_j \).

Let \( \mathcal{I} \) be the set of all non-ordered pairs of indices \( i \in \{1, \ldots, n\} \). To represent the fact that two agents can be on the same or on different paths, we partition \( \mathcal{I} \) in two sets \( \mathcal{I}_+ \) and \( \mathcal{I}_- \). The first set contains all pairs of indices of agents travelling along different paths, the second one contains all pairs of indices of agents travelling along the same path. The intersection is represented by an interval \([a_i, b_i]\) along the path of each agent, where the size of the interval must be chosen taking into account the geometry of the intersection and the physical size of the agent. A side impact occurs if two agents from different paths enter the interval \([a_i, b_i]\) simultaneously; a rear-end collision occurs if two agents on the same path have distance less than \( d \), fixed for all agents. The bad set
is the set of input signals. It has, however, a polynomial-time exact solution, first
proposed in [42], when all job durations \( p_i \) are identical. This case can always be transformed, by normalisation of the data, into the case where jobs have duration 1, denoted \( 1|\text{prec}|p_i=1|\text{Lmax} \).

One of the main results of the paper is a proof of equivalence of two decision problems. Here, equivalence is meant in the sense used in complexity theory [40]:

**Definition 2.** A problem \( P1 \) is reducible to a problem \( P2 \) if for every instance \( I \) of \( P1 \) an instance \( I' \) of \( P2 \) can be constructed in polynomial-bounded time, such that \( I \in P1 \) if and only if \( I' \in P2 \). In this case, we write \( P1 \prec P2 \). If \( P1 \prec P2 \) and \( P2 \prec P1 \) then we say that \( P1 \) and \( P2 \) are equivalent, denoted \( P1 \equiv P2 \).

### B. Function Spaces

Here we introduce three sets of input signals and prove some of their properties. The crucial properties are in Theorems 1 and 3. We attach a preorder “\( \prec \)” [43] to each set. We denote the maximum and minimum of a set in the preorder “\( \prec \)” by “\( \max_{\prec} \)” and “\( \min_{\prec} \)”, respectively. All proofs of this section are collected in Appendix A. From here until the end of Section IV, we deal with solutions of (1) for fixed initial conditions \((x(0),\dot{x}(0))\) and for fixed parameters \( \Theta \). Hence, for the sake of brevity, the dependence of all objects on \((x(0),\dot{x}(0))\) and \( \Theta \) is omitted.

The order of agents along a path imposes constraints on the order in which agents can reach the intersection. Considering the task of letting an agent through the intersection as a job to be executed on a machine, we represent these constraints as a set of precedence constraints on the jobs, writing \( J_i < J_j \) if job \( i \) must be executed before job \( j \), that is, if \( x_i(0) > x_j(0) \) for all \( i,j \in I \). The set of all precedence constraints defines a directed acyclic graph with the jobs as nodes, and a corresponding topological order (the order of the nodes along a directed path). We say that a vector \( T \in \mathbb{R}^n_+ \) has elements in topological order if \( T_i \leq T_j \) whenever \( J_i < J_j \). We say that agent \( j \) is the predecessor of agent \( i \) if there is such that \( J_j < J_i \) and \( J_k < J_j \) for all \( k \neq j \) such that \( J_k < J_j \). In other words, the predecessor of \( i \) is the agent immediately preceding \( i \) in the topological order. This is denoted \( J_j \prec J_i \). If \( i \) has no predecessor, we write \( 0 \prec J_i \). Given an assignment of indices to agents we say that these are numbered in topological order if, for all \( (i,j) \in I \), \( j < i \) implies \( J_j \prec J_i \), in reverse topological order if \( j < i \) implies \( J_i \prec J_j \).

**Definition 3.** \( \mathcal{U} \) is the set of input signals \( \mathcal{U} \) such that \((x(t,u),\dot{x}(t,u)) \notin B_- \). for all \( t \geq 0 \).

This is the set of all input signals that avoid rear-end collisions. We define the preorder “\( \mathcal{U} \)” on the components \( u_i \) of \( u \in \mathcal{U} \) by writing \( u_i \preceq u' \) if \( x_i(u_i) \leq x_i(u'_i) \) and we say that \( u \preceq u' \) if the above relation holds for all \( i \in \{1,\ldots,n\} \). \( \mathcal{U} \) has the following property.

**Theorem 1.** If \( \mathcal{U} \neq \emptyset \), then it has a minimum in the preorder “\( \mathcal{U} \)”.

---

**Fig. 2.** Three schedules for \( 1|\text{prec}|\text{Lmax} \) with \( \tau = [1.5,2,3,5] \), \( d = [6,6,5] \), \( p = [1.5,0.5,1] \), and \( J_2 < J_1 \). Schedule 1 is not feasible because \( J_1 \) terminates after its deadline. Schedule 2 is not feasible because the order of \( J_1 \) and \( J_2 \) violates a precedence constraint. Schedule 3 is feasible.

\( B \), consisting of all states where two or more agents collide, is the union of the set \( B_+ := \{ x \in \mathbb{R}^n : \exists (i,j) \in I_+, x_i \in (a_i,b_i) \) and \( x_j \in (a_j,b_j) \} \) which accounts for all side impacts, and \( B_- := \{ x \in \mathbb{R}^n : \exists (i,j) \in I_-, |x_i - x_j| < d \} \), which accounts for all rear-end collisions. Given the initial condition \((x(0),\dot{x}(0))\), an input signal \( u \) such that \((x(t,u),\dot{x}(t,u)) \notin B \) for all \( t \geq 0 \) is called a collision-free input. VP can be formally stated as follows.

**VP.** Given initial conditions \((x(0),\dot{x}(0))\) determine if there exists an input signal \( u \) that guarantees that \((x(t,u),\dot{x}(t,u)) \notin B \) for all \( t \geq 0 \).

An instance of VP is fully described by the initial condition \((x(0),\dot{x}(0))\) and by the tuple \( \Theta := \{a_1,\ldots,a_n,b_1,\ldots,b_n,d_1,\ldots,d_n,X_1,\ldots,X_n,U_1,\ldots,U_n,U\} \).

### III. Preliminary Results

In this section, we introduce some concepts of Scheduling Theory, define some function sets for the inputs of (1), and prove some properties of these sets that are needed in the following sections.

**A. Notions of Scheduling Theory**

A scheduling problem consists of assigning to a number of jobs a schedule, that is, a vector of execution times, while satisfying given constraints [28]. Job \( i \) is represented by the symbol \( J_i \). If a precedence constraint holds between two jobs, that is, if \( J_i \) must be executed before \( J_j \), we write \( J_i < J_j \). Adapting the formality of complexity theory, we write scheduling problems in the form of decision problems, which have a yes or no solution [40]. When a decision problem \( P \) maps an instance \( I \) to the solution yes we say that it accepts the instance, denoted \( I \in P \). In particular, the results presented here use a special form of the following decision problem:

**Definition 1.** \( 1|\text{prec}|\text{Lmax} \). Given a set of \( n \) jobs to be run on a single machine, with release times \( r_i \in \mathbb{R}_+ \), deadlines \( d_i \in \mathbb{R}_+ \), and durations \( p_i \in \mathbb{R}_+ \), and a set of precedence constraints, determine if there exists a schedule \( T = (T_1,\ldots,T_n) \in \mathbb{R}_+^n \) such that, for all \( i \in \{1,\ldots,n\} \), \( r_i \leq T_i \leq d_i - p_i \), and for all \( i \neq j \), \( T_i \geq T_j \Rightarrow T_i \geq T_j + p_j \), and \( J_i < J_j \Rightarrow T_i \leq T_j \).

Figure 2 shows three schedules for an instance of the above problem. \( 1|\text{prec}|\text{Lmax} \) was shown to be NP-Complete in [41]. It has, however, a polynomial-time exact solution, first
A minimum of $\mathcal{U}$ is a lowest control input that the agents can apply without causing a rear-end collision. Notice that, even though this minimum does not have to be unique, the corresponding position trajectory is unique. This is a simple consequence of the definition of “$\geq$”.

Assume that $\mathcal{U} \neq \emptyset$. Given $\mathbf{u} \in \max \mathcal{U}$, a vector $\mathbf{T} := (T_1, \ldots, T_n)$, and $\mathbf{x}(t) := \mathbf{x}(t, \mathbf{u})$, define the constraints

\begin{align*}
    x_i(0) &\leq a_i \Rightarrow x_i(t, u_i) \leq a_i \quad \forall t \leq T_i, \quad (5) \\
    J_j &\leq J_i \Rightarrow x_i(t, u_i) \leq x_j(t, u_j) - d \quad \forall t \geq 0, \quad (6) \\
    x_i(t, u_i) &\geq \underline{x}_i(t) \quad \forall t \geq 0. \quad (7)
\end{align*}

Constraint (5) requires that, if agent $i$ is behind $a_i$ at $t = 0$, then it remains behind $a_i$ at least until $t = T_i$. Constraint (6) requires that an agent and its predecessor respect a safety distance $d$. Constraint (7) requires that each agent maintain a trajectory not lower than the one obtained with the minimal input $\underline{x}_i$. Note that (7) is redundant when (6) is satisfied by all agents. It is nonetheless used in the definition of the input set $\mathcal{U}(x_j, T_i)$ which follows.

**Definition 4.** Given a vector $\mathbf{T} \in \mathbb{R}^n$ (a schedule), and given $\mathbf{x}(0)$, we let

- $\mathcal{U}(\mathbf{T})$ be the set of input signals $\mathbf{u} \in \mathcal{U}$ such that all components of $\mathbf{x}(t, \mathbf{u})$ satisfy (5) and (6);
- $\mathcal{U}(x_j, T_i)$ be the set of inputs $\mathbf{u} \in \mathcal{U}$ that satisfy (5), (6), and (7). We denote by $\mathcal{U}_i(\mathbf{0}, T_i)$ the same set when $i$ has no predecessor, and we write $\mathcal{U}_i(\cdot, T_i)$ when we need to refer to both cases. If $\mathcal{U} = \emptyset$, then $\underline{x}_i$ in (7) is not defined. In this case, let $\mathcal{U}_i(\cdot, T_i) := \emptyset$.

The set $\mathcal{U}(\mathbf{T})$ contains all the inputs that agents can use without violating (5) and (6). Notice that $\mathcal{U}(\mathbf{T}) \subset \mathcal{U}$, since any input satisfying (6) for all $i$ satisfies $\mathbf{x}(t, \mathbf{u}) \notin B$ for all $t \geq 0$. We attach a preorder to the above sets, as follows.

For all $\mathbf{u}, \mathbf{u}' \in \mathcal{U}_i(\cdot, T_i)$, we write $\mathbf{u} \preceq \mathbf{u}'$ if $x_i(t, u_i) \leq x_i(t, u_i') \forall t \geq T_i$, and we say that $\mathbf{u} \preceq \mathbf{u}'$ if the above relation holds for all $i \in \{1, \ldots, n\}$. We now address the existence of maxima of $\mathcal{U}(\mathbf{T})$ and, to do so, we first establish the existence of maxima of $\mathcal{U}_i(\cdot, T_i)$.

**Lemma 2.** Consider agents $i$ and $j$. If $J_j \leq J_i$, then take $u_j$ such that $u_j = u_j, \max$ for all $t \geq T_j$ for some finite $T_j \geq 0$, and consider the corresponding trajectory $x_j$. Assume $\mathcal{U} \neq \emptyset$ and $\mathcal{U}_i(\cdot, T_i) \neq \emptyset$. Then $\mathcal{U}_i(\cdot, T_i)$ has maxima in the preorder “$\preceq$”, and for every maximum $\mathbf{u}_i$ there exists a finite $T_i > 0$ such that $\mathbf{u}_i(t) = u_i, \max$ for all $t \geq T_i$.

Moreover, if $i$ has a predecessor $j$, then \( u'_j \preceq u_j, u_i \in \max_\preceq \mathcal{U}(x_j(u_j), T_i), u'_i \in \mathcal{U}_i(x_j(u_j), T_i) \Rightarrow u'_i \preceq u_i \).

Notice that, even though the maxima in Lemma 2 are not necessarily unique, the definition of the preorder “$\preceq$” implies that for all $u_i, u'_i \in \max_\preceq \mathcal{U}(\cdot, T_i)$, $x_i(t, u_i) = x_i(t, u'_i)$ for all $t \geq T_i$. Algorithm 1 computes a maximum of $\mathcal{U}_i(\cdot, T_i)$ considering elements of $\mathcal{U}_i(\cdot, T_i)$ of the form

\[
\begin{array}{lr}
    u_i \quad & \text{if } t < \min(t_1, t_2, t_3) \\
    u_i, \max & \text{if } t \in [\min(t_1, t_3), \min(t_2, t_3)] \\
    u_i, \min & \text{if } t \in [\min(t_2, t_3), t_3] \\
    u_i & \text{if } t > t_3.
\end{array}
\]

In the algorithm the function \( \text{Root}(F(\gamma)) \) returns a positive root of the equation \( F(\gamma) = 0 \). If \( \text{Root}(F(\gamma)) \) finds no positive solution the algorithm aborts although, for the sake of brevity, this is not stated explicitly in the pseudocode; in this case $\mathcal{U}_i(\cdot, T_i)$ is empty. We assume that if \( i \) has a predecessor \( j \), then \( \bar{u}_j \) has already been computed; this is ensured by processing agents in topological order.

**Algorithm 1** Compute a maximum of $\mathcal{U}_i(\cdot, T_i)$

\begin{enumerate}
    \item $\alpha \leftarrow \min(x_i(T_i, u_i, \max), a_i)$
    \item $\bar{t}_1 \leftarrow \text{Root}(F(\tau_1) := x_i(T_1, u_i, r_1, \infty, \infty) - \alpha)$
    \item $\bar{t}_2 \leftarrow \text{Root}(F(\tau_2) := x_i(T_1, u_i, 0, r_2, \infty) - \alpha)$
    \item if $\bar{t}_1$ has no predecessor $j$ then $\bar{t}_j \leftarrow \text{Root}(x_j(T_j) \geq x_i(t, u_i, r_2, \infty) + d \forall t \geq 0$ then
    \item $\bar{u}_i \leftarrow u_i, \bar{t}_1, \bar{t}_2, \bar{t}_j$
    \item else $t_{\bar{t}} \leftarrow \text{Root}(F(x_i(T_i)))$
    \item $\bar{t}_1 \leftarrow \text{Root}(F(\tau_1) := \min_{t \geq T_1} \{x_j(t, u_j) - x_i(t, u_i, T_1, \infty)\})$
    \item $\bar{t}_2 \leftarrow \text{Root}(F(\tau_2) := \min_{t \geq T_1} \{x_j(t, u_j) - x_i(t, u_i, T_2, \infty)\})$
    \item $\bar{t}_j \leftarrow \text{Root}(F(\gamma) := x_i(t, u_j) - x_i(t, u_i, r_j, \infty))$
    \item $\bar{u}_i \leftarrow u_i, \bar{t}_1, \bar{t}_2, \bar{t}_j$
    \item else
    \item $\bar{t}_1 \leftarrow 0$
    \item $\bar{t}_2 \leftarrow \text{Root}(F(\tau_2) := \min_{t \geq T_1} \{x_j(t, u_j) - x_i(t, u_i, T_2, \infty)\})$
    \item $\bar{t}_j \leftarrow \text{Root}(F(\gamma) := x_i(t, u_j) - x_i(t, u_i, r_j, \infty))$
    \item $\bar{u}_i \leftarrow u_i, \bar{t}_1, \bar{t}_2, \bar{t}_j$
    \item else
    \item $\bar{t}_1 \leftarrow 0$
    \item $\bar{t}_2 \leftarrow \text{Root}(F(\tau_2) := \min_{t \geq T_1} \{x_j(t, u_j) - x_i(t, u_i, T_2, \infty)\})$
    \item $\bar{t}_j \leftarrow \text{Root}(F(\gamma) := x_i(t, u_j) - x_i(t, u_i, r_j, \infty))$
    \item $\bar{u}_i \leftarrow u_i, \bar{t}_1, \bar{t}_2, \bar{t}_j$
    \item return $\bar{u}_i$
\end{enumerate}

**Theorem 3.** If $\mathcal{U}(\cdot) \neq \emptyset$, then it has a maximum in the preorder “$\preceq$”. Moreover, if the components of a vector $\mathbf{u} \in \mathcal{U}$ satisfy the recursive relation

\[
\begin{array}{c}
    0 \leq J_i \Rightarrow u_i \in \max_\preceq \mathcal{U}(\mathbf{0}, T_i), \\
    J_j \leq J_i \Rightarrow u_i \in \max_\preceq \mathcal{U}(x_j(t, u_j), T_i),
\end{array}
\]

then $\mathbf{u} \in \max_\preceq \mathcal{U}(\cdot)$.

The above result implies that the problem of finding a maximum of $\mathcal{U}(\cdot)$ has an optimal substructure [40], that is,
we can compute an optimal solution iteratively by solving a set of simpler optimization problems. Thus, finding a maximum of \( U(T) \) is no harder than finding a maximum of \( U_i(T_i) \). Here lies the keystone of the paper: once a feasible schedule \( T \) has been found, finding a maximum of \( U(T) \), and hence an input signal satisfying VP, is a simple task. The complexity of VP is entirely due to finding \( T \). This is the matter of the following section.

**IV. MAIN RESULTS**

**A. Formal Statement of SP and Equivalence of VP and SP**

In Section III, we defined the sets \( \mathcal{U} \) and \( U(T) \) and proved that they have a minimum and a maximum, respectively. Here, we use these minima and maxima to define the scheduling quantities \( R_i, D_i, \) and \( P_i \), that are necessary to formalise SP.

Let \( t_a(u_i) := \min \{ t \geq 0 : x_i(t, u_i) \geq a \} \), that is, \( t_a(u_i) \) is the earliest time when \( x_i \) becomes greater than or equal to \( a \). Such \( t_a(u_i) \) exists since \( \bar{x}_i \) is bounded and bounded away from 0. As before, let \( \mathbf{u} \) denote an element of \( \min \{ \mathcal{U} \} \). The scheduling quantities are defined as follows. If \( \mathcal{U} = \emptyset \), set \( R_i = 0, D_i = -1 \) for all \( i \in \{ 1, \ldots, n \} \), otherwise

\[
R_i := t_{a_i}(u_i, \text{max}), \\
D_i := t_{a_i}(u_i). 
\]

These are the earliest and latest time when an agent can reach \( a_i \). If \( \mathcal{U}(T) = \emptyset \), \( P_i(T) = \infty \) for all \( i \in \{ 1, \ldots, n \} \), otherwise, take \( \mathbf{u}(T) \in \max \mathcal{U}(T) \) and

\[
P_i(T) := t_b(u_i(T)). 
\]

P(T) is the earliest time when \( i \) can reach \( b_i \), avoiding rear end collisions, if it does not pass \( a_i \) before \( T_i \). We can now state:

**SP.** Given initial conditions \( \{ x(0), \dot{x}(0) \} \), determine if there exists a schedule \( T = (T_1, \ldots, T_n) \in \mathbb{R}^{n+} \) such that for all \( i \)

\[
R_i \leq T_i \leq D_i, 
\]

and for all \( (i, j) \in I_+ \), if \( x_i(0) < b_j \), then

\[
T_i \geq T_j \Rightarrow T_i \geq P_j(T). 
\]

Looking at the scheduling problem in Definition 1, we see that \( R_i \) plays the role of a release time, \( D_i \) plays the role of a deadline, and \( P_i - T_i \) plays the role of a job duration. The precedence constraints of Definition 1 are here a consequence of (13) and the definition of \( P_j \). The main difference with respect to a standard scheduling problem is that, here, the duration \( P_i - T_i \) is a function of the schedule \( T \). Notice that instances of Problems VP and SP are described by the same tuple \( \{ x(0), \dot{x}(0) \} \). According to Definition 2, this means that the mapping between instances of the two problems is the identity, and in order to prove equivalence it suffices to show that \( \{ x(0), \dot{x}(0) \} \in \mathcal{V} \) if and only if \( \{ x(0), \dot{x}(0) \} \in \mathcal{S} \). This is done in Theorem 5. In the proof we use the following Corollary of Theorem 3.

**Corollary 4.** Assume that \( \mathcal{U}(T) \neq \emptyset \). Then, \( P_i(T) \leq t_{b_i}(u_i) \) for all \( i \) and for all \( \mathbf{u} \in \mathcal{U}(T) \).

**Proof:** By Theorem 3, if \( \mathcal{U}(T) \neq \emptyset \) then it has a maximum \( \mathbf{u} \) in the preorder \( \triangleleft \). Given the definition of the preorder \( \triangleleft \), \( x_i(t, u_i) \geq x_i(t, u_i) \) for all \( u_i \in \mathcal{U}(T) \) and for all \( t \geq T_i \), which implies that \( t_{b_i}(u_i) = P_i(T) \leq t_{b_i}(u_i) \).

**Theorem 5.** VP \( \simeq \) SP.

**Proof:** We must show that VP is reducible to SP and vice versa, or equivalently that \( \{ x(0), \dot{x}(0), \Theta \} \in \mathcal{V} \Leftrightarrow \{ x(0), \dot{x}(0), \Theta \} \in \mathcal{S} \). We prove the two directions of the implication separately.

\[
\{ x(0), \dot{x}(0), \Theta \} \in \mathcal{V} \Rightarrow \{ x(0), \dot{x}(0), \Theta \} \in \mathcal{S}: \text{Assume that } \mathbf{x}(t, \mathbf{u}) \text{ satisfies the constraints of VP. The time instants at which } \mathbf{x}(t, \mathbf{u}) \text{ crosses each of the planes } x_i = a_i \text{ define a vector } \mathbf{T}, \text{ and we can set } T_i = 0 \text{ if } x_i(0) < a_i. \text{ Given this choice of } \mathbf{T}, \mathcal{U}(T) \neq \emptyset, \text{ and since } \mathcal{U}(T) \subseteq \mathcal{U} \neq \emptyset, \text{ so by Theorem 1 the quantities } x_i(t) \text{ exist. Moreover this } \mathcal{T} \text{ satisfies (12) given the definition of } R_i \text{ and } D_i. \text{ The time instants at which } \mathbf{x}(t, \mathbf{u}) \text{ crosses the planes } x_i = b_i \text{ define a vector } \mathbf{P} = (P_1, \ldots, P_n), \text{ with } P_i = t_{b_i}(u_i), \text{ and for all } i \text{ such that } T_i > T_j, (i, j) \in I_+ \text{ and } x_i(0) < b_i, \text{ we have } T_i > P_j, \text{ otherwise } \mathbf{x}(t, \mathbf{u}) \in \bigotimes \neq \emptyset. \text{ By Corollary 4, } P_i(T) \leq t_{b_i}(u_i) = P_i(T) \text{ for all } u \in \mathcal{U}(T), \text{ so we conclude that } \mathbf{T} \text{ satisfies (13)}.
\]

Thus, together with the fact that all inputs in \( \mathcal{U}(T) \) satisfy (5) and (6) and that \( \mathbf{x}(t) > \bar{x}_{i, \text{min}} > 0 \) for all } t, ensures that: (i) if \( x_i(0) \leq a_i, \text{ then } x_i(t, u_i) \leq a_i \text{ for all } t < T_i, \) (ii) if \( x_i(0) < b_i \), then \( x_i(t, u_i) < b_i \text{ for all } t < P_i(T), \) and (iii) for all \( t > 0 \), \( x_i(t, u_i) \leq x_j(t, u_j) - d \text{ when } J_j < J_i \). Given (12) and (13) with this input each \( x_i \) enters the interval \((a_i, b_i)\) no earlier than \( T_i \), leaves the interval at \( P_i(T) \), and the intervals \((T_i, P_i(T))\) and \((T_j, P_j(T))\) with \((i, j) \in I_+ \) do not intersect. Thus \( \mathbf{x}(t, \mathbf{u}) \notin \bigotimes \) for all \( t \geq 0, \) and \( \mathbf{u} \) satisfies VP.

**B. Exact Solution of VP**

By Theorem 5 a solution of VP, which is defined over a function space, can be found by solving SP, whose search space is the set of all the possible orderings of agents through the intersection, and is therefore finite. Here we propose an enumerative solution of SP, and we start by further reducing the size of the search space by means of the following Lemma.

**Lemma 6.** If SP accepts an instance, then it is satisfied by a schedule \( T \) with elements \( T_i \), in topological order, that is, such that \( J_j < J_i \), constraint (5), appearing in the definition of \( P_i \), is inactive for all \( T_i \leq T_j \).

**Proof:** The property follows from the fact that, if \( J_j < J_i \), constraint (5), appearing in the definition of \( P_i \), is inactive for all \( T_i \leq T_j \).
This result enables us to restrict the search to schedules in topological order. Furthermore, given a candidate order of jobs $J_i$ in topological order, if a schedule is feasible, then in particular the one that satisfies (13) tightly is feasible, i.e., the one such that $T_i \geq T_j \Rightarrow T_i = P_j(T)$. Such a schedule is uniquely determined once the order of jobs has been chosen. Therefore, the search space coincides with the set $P$ of all possible permutations of the indices $1, \ldots, n$ that satisfy the topological order ($J_i < J_j \Rightarrow j < i$). The procedure EXACTSOLUTION in Algorithm 2 solves SP exactly by performing an exhaustive search in $P$. We denote by $\pi \in P$ a permutation of indices and by $\tau_i$ the $i$-th index in the permutation. With some abuse of notation, in the procedure we write $\bar{P}_i \leftarrow b_{\pi_i}(\max_{U_i} U_i(\cdot, \tau_i))$ as a short form of $u_i \in \max_{U_i} U_i(\cdot, \tau_i)$. The abbreviation should cause no confusion, since as we noted before, all maxima of $\max_{U_i} U_i(\cdot, \tau_i)$ give the same trajectory for $t \geq T_i$, and therefore the same value of $t_{b_i}$. The state $(\bar{x}(0), \dot{x}(0))$ is in with $\dot{x}_{i,\min} = 1$, $\dot{x}_{i,\max} = 10$, $u_{i,\min} = -1$, $u_{i,\max} = 1$, $a_i = 5$, $b_i = 6$, and $d = 1$. Assume $x_1(0) = 0$, $x_2(0) = 1$, $x_3(0) = 0$, and $\dot{x}(0) = 1$ for all agents, and let agents 1 and 2 be on the same path. The initial conditions imply that $J_2 < J_1$.

We have that $u_i(t) = -1$ for all $t \geq 0$, for all $i$. Assume that Algorithm 2 is testing the permutation $[2, 1, 3]$. The values of $R_i$ and $D_i$ found using $u_{i,\max}$ and $u_2$ computed above are, respectively, $R_i \approx [2.317, 2.317]$ and $D_i = [5, 4.5]$. Lines 5-7 of Algorithm 2 assign $T_2 = 2$, $u_2 = 1$, $\bar{P}_2 \approx 2.317$. Then, agent 1 has agent 2 as predecessor. Line 10 of Algorithm 2 assigns $T_1 = 2$. To compute the assignment at Lines 15 and 16, we use Algorithm 1. All roots in Algorithm 1 can be found analytically, since the vector field can be integrated explicitly. We find that $\bar{\tau}_1 = 0$ and $x_2(t, 1) \geq x_1(t, u_{1,0,\infty})$, so that the assignment at Line 5 of Algorithm 1 sets $u_1 = 1$ for all $t \geq 0$. This gives $\bar{P}_1 \approx 2.606$. Finally, Lines 12, 21, and 22 of Algorithm 2 assign $T_3 \approx 2.606$, $u_3(t) = 0$ for $t \in [0, 0.418]$, $u_3(t) = u_{3,\max}$ for $t > 0.418$, and $\bar{P}_3 \approx 2.488$. We thus obtain $T = [2, 2, 2.606]$, which verifies the test at Line 25.

The search space, and hence the running time of the procedure EXACTSOLUTION in Algorithm 2, scales as the multinomial coefficient $(n_1, n_2, \ldots) = n!/n_1!n_2!\ldots$ where $n_i$ is the number of agents on path $i$ and $\sum_i n_i = n$. For an effective, online approach applicable to large systems we need to approximate an exact solution.

C. Approximation of Solution of VP

The results in the previous section provide a means to determine membership in the maximal controlled invariant set exactly. However, the complexity of the algorithm renders the computation impractical in the presence of a large number of agents. Here, we exploit the equivalence of VP and SP to construct an approximate solution with polynomially bounded running time, and we provide an upper bound for the error introduced by the approximation. Specifically, we use Garey’s exact solution [42] of 1 $\tau_1$, $\tau_2, \tau_3 = 1$ $\tau_{\max}$ (this is the scheduling problem of Definition 1 with unit time jobs) to solve an approximate version of SP. The proofs of this section are collected in Appendix B. Let POLYNOMIALTIME be a procedure that solves 1 $\tau_1$, $\tau_2, \tau_3 = 1$ $\tau_{\max}$. The idea is to define a time $\delta_{\max}$ long enough so that any agent is able to cross the interval $[a_i, b_i]$ in at most $\delta_{max}$, and allocate this fixed amount of time to each agent.

To begin with, consider the quantity

\[ d^*_i := \inf \left\{ \alpha : \exists (u_i, u_j), x_i(u_i) + d \leq x_j(u_j), \right\} \]

This is the minimum distance that two agents on the same path must have, at any given time instant, to avoid a rear end collision, if the agent in front has velocity $\dot{x}_{j,\min}$ and the one in the back has velocity $\dot{x}_{i,\max}$. Then, call

\[ \tau_i(\alpha) := \inf \{ t \geq 0 : x_i(t, u_{i,\min}) \geq \max\{b_i, a_i + d^*_i\}, \right\} \]

(14)
This is the minimum time taken to reach $\max\{b_i, a_i + d_t^i\}$ by an agent that starts at position $\alpha$ with minimum velocity, using input $u_{t,\text{max}}$. Finally, call

$$\delta_{\text{max}} := \max_{i \in \{1, \ldots, n\}} \tau_i(u_i).$$

(15)

We can now define the approximation $\text{SP}^\ast$ of $\text{SP}$:

**SP**. Given initial conditions $(x(0), \dot{x}(0))$, determine if there exists a schedule $T = (T_1, \ldots, T_n) \in \mathbb{R}^n_+$ such that, for all $i$,

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

(16)

for all $(i, j) \in I_+$, if $T_j = 0$ and $x_j(0) < b_i$ then

$$T_i \geq T_j \Rightarrow T_i \geq P_j(T),$$

(17)

for all $(i, j) \in I_-$ if $T_j = 0$ then

$$J_j < J_i \Rightarrow T_i \geq \tau_j(x_j(0))$$

(18)

while for all $(i, j) \in I$ if $T_j > 0$ then

$$T_i \geq T_j \Rightarrow T_i \geq T_j + \delta_{\text{max}}$$

(19)

and

$$J_j < J_i \Rightarrow T_i \geq T_j,$$

(20)

Note that, by (16), $T_j = 0$ if and only if $x_j(0) \geq a_j$. Constraints (16) and (17) are the same as in $\text{SP}$. Constraint (18) states that, if $i$ and $j$ are on the same path and $j$ lies at or after $a_j$, then agent $i$ is not allowed in the intersection before $j$ has passed the points $b_j$ and $a_j + d_t^j$. Constraint (19) states that, if $i$ and $j$ lie before $a_i$ and $a_j$, respectively, then their scheduled time of arrival must be spaced at least $\delta_{\text{max}}$ apart. Finally, Constraint (20) requires that schedules be in topological order. An exact solution to the above problem is found by Algorithm 3. In the pseudocode, without loss of generality, we assume that $x_i(0) \geq a_i$ for $i = 1, \ldots, m$, and $x_i(0) < a_i$ for $i = m + 1, \ldots, n$. $P_j(0)$ at Line 7 stands for $P_j(T)$ with $T_i = 0$ for all $i$. $\text{Algorithm 3 Solution of SP}^\ast$

1: procedure $\text{APPROXIMATE SOLUTION}(x(0), \dot{x}(0), \Theta)$
2: for all $i \in \{1, \ldots, m\}$ do
3: if $x_i(0) \in [a_i, b_i)$ and $x_i(0) \in [a_i, b_i)$ for some $(i, j) \in I_+$ or $D_i < R_i$ for some $i$ then
4: return $\emptyset$ (no)
5: for all $i \in \{1, \ldots, m\}$ do
6: if $i \in \{m + 1, \ldots, n\}$ do
7: \(R_i \leftarrow \max_{\{j \leq m \mid i \neq j\}} \{P_j(0)\}, \max_{\{j \leq m \mid i \neq j\}} \{\tau_j\}, R_i\}
8: set $\delta_{\text{max}}$ as in (15)
9: $r = (R_{\text{max} + 1} / \delta_{\text{max}}, \ldots, R_{\text{max}} / \delta_{\text{max}})$
10: $d = (D_{\text{max} + 1} / \delta_{\text{max}}, \ldots, D_{\text{max}} / \delta_{\text{max}} + 1)$
11: $\Theta_{n+1} = \text{POLYNOMIAL TIME}(r, d, \text{prec})$
12: for $i = m + 1$ to $n$ do
13: $T_i \leftarrow r_i$, $\delta_{\text{max}}$
14: return $\Theta$ (answer)

Example 2. Consider the system in Example 1, but let $x_1(0) = 0$, $x_2(0) = 4$, $x_3(0) = 0$, and let $a_i = 30, b_i = 31$, $d = 1, \dot{x}(0) = 1$ for all agents. We can compute $d_t^i$ and $\delta_{\text{max}}$, explicitly, obtaining $d_t^i = 21.25$ and $\delta_{\text{max}} \approx 0.37$. All agents have $x_i(0) < a_i$, therefore $R_i = R_{\text{max}}$ in Algorithm 3. By proceeding as in Example 1 we obtain $R = R_{\text{max}} \simeq [7.62, 7.07, 7.62]$. and $D = [30, 26, 30]$. The vectors $r$ and $d$ obtained by dividing $R$ and $D$ by $\delta_{\text{max}}$ are $r \simeq [2.11, 1.11, 1.19]$ and $d \simeq [4.71, 4.08, 4.71]$. The procedure $\text{POLYNOMIAL TIME}$ finds the feasible schedule $T \simeq [3.11, 1.11, 2.11]$, which corresponds to $T \simeq [19.8, 7.07, 13.43]$.

We say that $\text{SP}^\ast$ is an approximation of $\text{SP}$ if any schedule that is feasible for $\text{SP}^\ast$ is feasible also for $\text{SP}$.

**Theorem 7.** $\text{SP}^\ast$ is an approximation of $\text{SP}$.

By this theorem, $\text{APPROXIMATE SOLUTION}$ can be used to check membership in the maximal controlled invariant set, but it underestimates its size. The extent of this underestimation depends on the choice of $\delta_{\text{max}}$. The following theorem provides a measure of the extent of the underestimation. First, define the sets $\hat{B}_+ := \{x : \exists (i, j) \in I_+, x_i \in [a_i, a_j + \delta_{\text{max}} \cdot \dot{x}_{\text{max}}], x_j \in [a_j, a_j + \delta_{\text{max}} \cdot \dot{x}_{\text{max}}]\}$ and $\hat{B}_- := \{x : \exists (i, j) \in I_-, |x_i - x_j| \leq \delta_{\text{max}} \cdot \dot{x}_{\text{max}}\}$ and define the extended bad set

$$\hat{B} := \hat{B}_+ \cup \hat{B}_-.$$  (21)

The extended bad set is thus a superset of the bad set. In Theorem 8 we prove that if $\text{APPROXIMATE SOLUTION}$ returns “no” then, for all $u \in U$ trajectories intersect $\hat{B}$ (i.e., $\hat{B}$ is an upper bound of the overestimation of $B$), then in Theorem 9 we prove that $\hat{B}$ cannot be made any smaller (i.e., $\hat{B}$ is a tight upper bound).

**Theorem 8.** If for a given $(x(0), \dot{x}(0))$ $\text{APPROXIMATE SOLUTION}$ returns “no”, then for all $u \in U$ there exists a $t \geq 0$ such that $x(t, u) \in \hat{B}$.

Now define the sets $\hat{B}_+ := \{x : \exists (i, j) \in I_+, x_i \in [a_i, a_j + \gamma_+ \delta_{\text{max}} \cdot \dot{x}_{\text{max}}], x_j \in [a_j, a_j + \gamma_+ \delta_{\text{max}} \cdot \dot{x}_{\text{max}}]\}$ and $\hat{B}_- := \{x : \exists (i, j) \in I_-, |x_i - x_j| \leq \gamma_- \delta_{\text{max}} \cdot \dot{x}_{\text{max}}\}$ and let $\hat{B} := \hat{B}_+ \cup \hat{B}_-.$

**Theorem 9.** If $\gamma_+ < 1$ or $\gamma_- < 1$, there exists a tuple $(x(0), \dot{x}(0), \Theta)$ such that $\text{APPROXIMATE SOLUTION}$ returns “no” and, for at least one $u \in U$, $x(t, u) \notin \hat{B}$ for all $t \geq 0$.

V. SYNTHESIS OF A SAFETY-ENFORCING SUPERVISOR

Here, we use the results of the previous sections to construct a supervisor for (1) to keep the system within the maximal controlled invariant set.

Given a discretization of time of step $\tau$, we design the supervisor as a map $s(x(k\tau), \dot{x}(k\tau), v_k) \mapsto u_k$ for all $k \in \mathbb{N}$. The map takes a “desired” input $v_k \in U := \{U_1, \ldots, U_n\}$ for the time interval $[k\tau, (k+1)\tau)$, and returns a signal $u_k(t) = v_k$ for all $t \in [k\tau, (k+1)\tau)$ if this maintains the state of the system within the maximal controlled invariant set, or a safe signal otherwise. Since all quantities here are evaluated at multiple time steps, the initial state $(x(k\tau), \dot{x}(k\tau))$ is explicitly included among the arguments of the trajectories. Let $\nu_k$ denote the desired input at time $k\tau$. Consider the two signals $u_k$ and $\hat{u}_k$ defined as follows: the first one is defined on the interval $[k\tau, (k+1)\tau]$ and identical to $\nu_k$; the second one is an element of $U$ defined on $[k\tau, \infty)$, and such that $u_k(t) = u_k(t)$ when $t \in [k\tau, (k+1)\tau]$. Additionally, given $(x(k\tau), \dot{x}(k\tau))$, call $u_{k, \text{safe}} \in U$ a control signal
such that $x(t, u_{k,safe}, x(κ), ˙{x}(κ)) \notin B$ for all $t \geq kτ$ (if such control exists), and call $u_{k,safe}$ the restriction of $u_{k,-safe}$ to the interval $[κ, (κ+1)τ]$. If $u_{k,-safe}$ does not exist, let $u_{k,-safe} = \emptyset$. The supervisor design problem is formally stated as follows

**Problem 1.** Design a supervisor $s(x(κ), ˙{x}(κ), v_K)$ such that

$$s(x(κ), ˙{x}(κ), v_K) = \begin{cases} u_k & \text{if } \exists u_k \in U : x(t, u_k, x(κ), ˙{x}(κ)) \notin B \\
\emptyset & \text{otherwise.} \end{cases}$$

The above supervisor overrides the desired input $v_K$ whenever this will cause a collision at some future time. Moreover, it has the following property.

**Proposition 10.** The supervisor $s(x(κ), ˙{x}(κ), v_K)$ defined above is nonblocking, i.e., if $u_k := s(x(κ), ˙{x}(κ), v_K) \neq \emptyset$ and $x_{k+1} = x(τ, u_k, x(κ), ˙{x}(κ)), ˙{x}_{k+1} = ˙{x}(τ, u_k, x(κ), ˙{x}(κ))$, then for any $v_K$, $s(x_{k+1}, ˙{x}_{k+1}, v_K) \neq \emptyset$.

**Sketch of proof:** Nonblockingness follows from the associativity property of the flow action: if at time $kτ$ there is a trajectory that does not intersect $B$, and this trajectory is followed for a time $τ$, then the same trajectory is available at time $(k+1)τ$.

This ensures that at all times $κτ$, and for all desired inputs $v_K$, the supervisor returns a collision-free input.

Given a system of the form (1) and the state $(x(κ), ˙{x}(κ))$ at some time $κτ$, the procedure `EXACT SOLUTION` returns a binary value (yes/no) and a schedule $T$. We can use this information to design the supervisor in Problem 1. Assume that, at $t = 0$, we have `EXACT SOLUTION`(x(0), ˙{x}(0), Θ) = {T₀, yes}. Define $u_{k, safe} = u(x(0), ˙{x}(0), T₀)$, where $u(x(0), ˙{x}(0), T₀) \in U(T₀)$, and we have explicitly written the initial conditions for clarity. Define $u_{k, safe}$ as the restriction of $u_{k, safe}$ to the time interval $[0, τ]$. At each iteration $k = 0, 1, 2, \ldots$, the supervisor map $s(x(κ), ˙{x}(κ), v_K)$ is defined in Algorithm 4.

**Algorithm 4 Implementation of the supervisor map**

1: procedure `s(x(κ), ˙{x}(κ), v_K)`
2: \( u(0) = v_K \) \( \forall t \in [κ, (κ+1)τ] \)
3: \( x_{k+1} = x(κ+1)τ, u(x(κ), ˙{x}(κ)) \)
4: \( x_{k+1} = x(κ+1)τ, u(x(κ), ˙{x}(κ)) \)
5: \( \{T, answer\} \leftarrow \text{EXACT SOLUTION}(x_{k+1}, ˙{x}(κ+1), Θ) \)
6: if \( answer = yes \) and \( x(t, u) \notin B \) for all \( t \in [κ, (κ+1)τ] \)
7: then
8: \( return \ u \)
9: else
10: \( \{T, answer\} \leftarrow \text{EXACT SOLUTION}(x(κ), ˙{x}(κ), Θ) \)
11: \( u_{k, safe} = u(x(κ), ˙{x}(κ), T) \)
12: \( return u_{k, safe} \)

**Theorem 11.** Assume that $s(x(0), ˙{x}(0), v_K) \neq \emptyset$. Then, the supervisor defined by Algorithm 4 solves Problem 1.

**Proof:** This follows from the equivalence of VP and SP.

Algorithm 4 uses `EXACT SOLUTION`, whose running time scales polynomially with the number of agents. Therefore, it can be applied only to relatively small problems. To achieve a supervisor that scales polynomially with the number of controlled agents, we substitute `EXACT SOLUTION` with `APPROXIMATE SOLUTION` at lines 5 and 9 of the Algorithm. Through this substitution we obtain a supervisor, denoted $s_{approx}(x(κ), ˙{x}(κ), v_K)$, that can handle much larger systems, at the expense of a more restrictive behaviour. The following result quantifies the restrictiveness of $s_{approx}$. Consider the extended bad set $B$ defined in (21). Call $\hat{s}(x(κ), ˙{x}(κ), v_K)$ the supervisor defined in Problem 1 substituting $\hat{B}$ to $B$.

**Theorem 12.** $s_{approx}(x(κ), ˙{x}(κ), v_K)$ is no more restrictive than $\hat{s}(x(κ), ˙{x}(κ), v_K)$, that is, if $s_{approx}(x(κ), ˙{x}(κ), v_K) = u_{k, safe}$ then $\hat{s}(x(κ), ˙{x}(κ), v_K) = u_{k, safe}$. Moreover if $s_{approx}(x(0), ˙{x}(0), v_K) \neq \emptyset$ then $s_{approx}$ is nonblocking in the sense defined in Proposition 10.

**Sketch of proof:** The fact that $s_{approx}(x(κ), ˙{x}(κ), v_K)$ is no more restrictive than $\hat{s}(x(κ), ˙{x}(κ), v_K)$ is a consequence of Theorem 8. Nonblockingness is proved by observing that, at iteration $k$, line 6 of Algorithm 4 explicitly checks the existence of a collision-free input for iteration $k+1$. Thus the only way that the algorithm could block is when the test at line 6 fails, the algorithm executes the code from line 8, and line 9 does not return a feasible schedule. This never happens for the following reason. If at iteration $k-1$ the desired input was accepted (i.e. the test at line 6 didn’t fail), then existence of a feasible schedule has already been verified. Otherwise at iteration $k-1$ agents have been forced to use input $u_{k-1, safe}$. Given the trajectory $x(t, u_{k-1, safe})$, let $T_{k-1}$ be the vector of times at which agents starting at $x(k-1)τ$ enter the intersection, with $T_{k-1} = 0$ if $x_i(k-1)τ ≥ a_i$. Let $T_k$ be the same vector computed from $x(κ, u_{k, safe})$. On can check that if $T_{k-1}$ satisfies (16)-(20) then so does $T_k$, therefore line 9 always finds a feasible schedule.

Thus, the supervisor $s_{approx}$ enforces safety, has polynomial complexity in the number of agents, and its performance can be rigorously compared against that of the optimal supervisor.

**VI. EXAMPLE**

We have tested the supervisor algorithms described in Section V on a set of vehicles governed by the equation

$$\bar{x}_i = \begin{cases} u_i - 0.005(\bar{x}_i) & \text{if} \\
(\bar{x}_i > \bar{x}_{i, min} \text{ and } u - 0.005(\bar{x}_i) \leq 0) \text{ or} \\
(\bar{x}_i < \bar{x}_{i, max} \text{ and } u - 0.005(\bar{x}_i) \geq 0) \\
0 & \text{otherwise,} \end{cases}$$

where the input term $u_i$ is the net effect of the motor’s torque and rolling resistance on the vehicle acceleration, while the quadratic term accounts for air drag [44]. The above system satisfies Assumptions (A.1)-(A.6). We have used the following parameters for all vehicles: $b_i = a_i = 10m$, $x_{i, max} = 13.9m/s$, $x_{i, min} = -2m/s^2$, $u_{i,max} = 2m/s^2$, $d = 5m$. This gives $b_i^2 ≈ a_i + 16.9989m$ and $d_{max} ≈ 3.6086s$.

Algorithm 4 was implemented numerically, as explained in [45], with $τ = 0.2s$. Trajectories were discretized with time step 0.1s. All simulations were executed on a 2.4GHz Intel core 2 Duo, 4Gb ram. Figure 3 shows a portion of the capture.
set (complement of the maximal controlled invariant set) in the space of the positions, for fixed velocities, for 3 agents on 3 paths. In Fig. 4 we show the trajectories of three pairs of agents travelling along 3 paths. In the top panel we use the supervisor with procedure EXACTSOLUTION. Notice that agents on different paths (trajectories in different colours) enter the intersection (in gray) as soon as the preceding agent has left it. EXACTSOLUTION was executed, in this configuration, in less than 0.38s. The bottom panel shows the same system and initial conditions, controlled using procedure APPROXIMATESOLUTION. APPROXIMATESOLUTION was executed, in this configuration, in less than 0.028s. Finally, Fig. 5 depicts the trajectories of 30 vehicles moving along 3 different paths, controlled using procedure APPROXIMATESOLUTION. APPROXIMATESOLUTION was executed in less than 0.17s.

Fig. 3. The capture set (the complement of the maximal controlled invariant set) for 3 agents on 3 different paths. Initial velocities are [5, 12, 3].

Fig. 4. Supervisor of 6 agents on 3 paths, $\dot{x}_i(0) = 13.9$ for all agents. The gray band marks the interval $[a_i, b_i]$, identical for all agents. Trajectories of agents on the same path are plotted in the same colour. Red segments on the $t$ axis denote intervals where the supervisor is overriding the desired input.

VII. CONCLUSION

We have proved that checking membership in the maximal controlled invariant set for $n$ dynamic agents at the intersection of $m$ paths is equivalent to solving a scheduling problem. By means of this equivalence, we have devised exact and approximate solutions to design least restrictive supervisors for collision avoidance at traffic intersections. These algorithms determine if a point $x$ in the state space belongs to the maximal controlled invariant set, that is, if there exists a conflict-free trajectory starting at $x$. The exact algorithm exploits the equivalence to reduce the verification problem to a search over a finite set of strings, corresponding to all possible ordering of agents. It has combinatorial complexity, which limits its applicability, but it represents the benchmark against which the performance of any solution can be tested. The running time of the approximate algorithm scales polynomially in the number of agents. Moreover, we can provide tight upper bounds on the error introduced by the approximation. We have proposed a least restrictive supervisor based on the exact solution, and an approximate supervisor with quantified approximation bounds. We have tested our results on a nonlinear model of vehicles at traffic intersections.

We are investigating extensions of the Scheduling Problem that we have introduced to handle more complex traffic scenarios, including merging paths, more complex intersection topologies, imperfect information, and model uncertainty. Some preliminary results have been published in [46].

APPENDIX A

PROOFS OF SECTION III

We begin by listing a few properties of the solutions of (1). In the following propositions and in the proof of Theorem 1 we consider only agents on a single path, therefore for simplicity of notation we omit the subscript $i$ from the quantities $u_{i, max}$ and $u_{i, min}$. The first two propositions relate the ordering of the extrema of segments of two trajectories with the ordering of the interior points of the segments.

Proposition 13. Let $u_i', u_i'' \in U_i,$ and let $u_i'(t) := u_{i, max}$ for all $t \in [t_1, t_2]$. Then $x_i(t_1, u_i'') \leq x_i(t_1, u_i'), x_i(t_2, u_i'') \leq x_i(t_2, u_i') \Rightarrow (x_i(t, u_i'') \leq x_i(t, u_i') \forall t \in [t_1, t_2]).$

Proposition 14. Let $u_i', u_i'' \in U_i,$ and let $u_i'(t) := u_{i, min}$ for all $t \in [t_1, t_2]$. Then $x_i(t_1, u_i'') \leq x_i(t_1, u_i'), x_i(t_2, u_i'') \leq x_i(t_2, u_i') \Rightarrow (x_i(t, u_i'') \leq x_i(t, u_i') \forall t \in [t_1, t_2]).$

Propositions 13 and 14 are simple consequences of the monotonicity property (4).

Proposition 15. Let $u_{i, \alpha} \in U_i, u_{j, \beta} \in U_j$ be two families of inputs, parametrised in $\alpha \in A$ and $\beta \in B$, and assume that there exists a finite $t_s \geq t_0$ such that $u_{i, \alpha}(t) = u_{i, max}$ and $u_{j, \beta}(t) = u_{j, min}$ for all $t \geq t_s$ and for all $\alpha \in A, \beta \in B.$
Assume that \( x_i(t, u, o) \) and \( x_j(t, u, \beta) \) depend continuously on the parameters \( \alpha \in A \) and \( \beta \in B \), where \( A \) and \( B \) are path connected. Let \( i, j \) be two agents with identical dynamics, and take \( \gamma > 0 \). If there exists a pair \((\alpha', \beta') \in A \times B \) such that \( x_i(t, u, \alpha') > x_j(t, u, \beta') + \gamma \) for all \( t \geq 0 \), and a pair \((\alpha'', \beta'') \in A \times B \) such that \( x_i(t, u, \alpha'') < x_j(t, u, \beta'') \) for some finite \( t \), then there exists a pair \((\alpha''', \beta''') \in A \times B \) such that \( x_i(t, u, \alpha''') \geq x_j(t, u, \beta''') + \gamma \) for all \( t \geq 0 \) and \( x_i(t, u, \alpha''') \) is tangent to \( x_j(t, u, \beta''') + \gamma \) at some finite \( t \).

Proof: Given a path from \((\alpha', \beta') \) to \((\alpha'', \beta'') \) in parameter space, (A.5) ensures that \( x_i(t, u, o) > x_j(t, u, \beta) - \gamma \) for all \( t \geq t_f \) for some \( t_f \) sufficiently large, for all pairs \((\alpha, \beta) \) along the path. Then, the intermediate value theorem applied to the function \( \min_{u \in (0, t_f)} \{ x_i(t, u, o) - x_j(t, u, \beta) - \gamma \} \) of \((\alpha, \beta) \) ensures the existence of a pair \((\alpha''', \beta''') \) such that \( \min_{i, \alpha} \{ x_i(t, u, \alpha''') - x_j(t, u, \beta''') - \gamma \} = 0 \). Differentiability of \( x_i(t, u, \alpha''') \) and \( x_j(t, u, \beta''') \) in \( t \) implies that they are tangent at some \( t \in (0, t_f) \).

The following continuity property of the solutions of (1) is a consequence of (A.2) and (A.4).

**Lemma 16.** For every \( \epsilon > 0 \) there exists a \( \delta > 0 \) such that, if \( |x_k(t) - x_{k'}(t)| < \delta \) for all \( t \in [0, \tau] \), then \( |x(t, u, \alpha''', \tau_{n}) - x(t, u, \alpha'', \tau_{n})| < \epsilon \) for all \( t \in [0, \tau_{n}] \).

We can now prove Theorem 1.

**Proof of Theorem 1:** The proof is constructive. Since the inequality conditions defining \( B_\infty \) only involve pairs of agents on the same path, the set \( \bar{U} \) is the cross product of a number of sets equal to the number of paths, each set containing the same curves, and the corresponding parameters \( \tau_1 \) and \( \tau_2 \) are represented in different shades of gray.

**Sketch of proof of Lemma 2:** We need to prove that 

\[
\begin{align*}
(1) \quad \exists \bar{u} \in \bar{U} \cap (1, T) : u_i \preceq \bar{u} \forall u_i \in U_i(1, T), \\
(2) \quad \left\{ \begin{array}{l}
J_i \ll J_i, u_i' \preceq u_i \in \bar{U} \cup_t x_j(u_j(T_i), T_i), u_i' \subseteq \bar{U}(x_j(T_i), T_i) \\
(3) \quad \bar{u}_i(t) \in u_i, \max \text{ for all } t \geq t_f > 0.
\end{array} \right.
\end{align*}
\]

In the case \( x_i(0) \leq a_i \), we introduce the family of inputs \( U_{b+c} \) represented in Fig. 6, which have the form

\begin{equation}
\begin{aligned}
\bar{u}_i := \left\{ \begin{array}{l}
u_i \forall t \in [0, \tau_1], \\
u_i, \max \forall t \in [\tau_1, \tau_2], \\
u_i, \min \forall t > \tau_2
\end{array} \right. (23)
\end{aligned}
\end{equation}

with \( \tau_1 \in [0, \tau_2] \) and \( \tau_2 \in [0, \infty] \). \( U_{b+c} \) is such that \( \tau_1 \leq T_1 \leq T_2 \) and the corresponding trajectories reach \( a_i \) exactly at \( T_i \). \( U_{b+c} \) is such that \( \tau_1 \leq \tau_2 \leq T_1 \) and the corresponding trajectories reach \( a_i \) exactly at \( T_1 \). \( U_{b+c} \) is such that \( \tau_1 \leq \tau_2 \leq T_1 \) and \( x_i(T_i, u_i) \leq a_i \). It can be proven that the above family satisfies:

(1) it is totally ordered and has a maximum \( (\tau_1 \leq \tau_2, \tau_2 = \infty) \) and a minimum \( (\tau_1 = \tau_2 = 0) \) in the preorder "\( \preceq \)";

(2) \( \min_{b+c} U_{b+c} \geq \bar{u}_i \).
To prove the above property, let us number and analyze agents in topological order. Let \( \{1, \ldots, m\} \) be all agents such that \( x_i(0) \geq a_i \), and let

\[
x_i^* := x_i(u_{i,\text{max}}, a_i, \hat{x}_i, \min),
\]

that is, \( x_i^* \) is a trajectory of agent \( i \) with maximum input, initial position equal to \( a_i \), and initial velocity equal to \( \hat{x}_i,\min \). If \( \mathcal{U}^* \) admits a schedule then \( \mathcal{U}^* \neq \emptyset \), otherwise condition (16) could not be satisfied. Using this fact and Proposition 15 we can construct an input \( u \in \mathcal{U} \) of the form

\[
u_i(t) = \begin{cases} u_{i,\text{max}} & \text{for } t \leq t_1 \\ u_{i,\text{min}} & \text{for } t \in (t_1, t_2] \\ u_i & \text{for } t > t_2 \end{cases}
\]

with \( 0 \leq t_1 \leq t_2 \leq \infty \) and \( J_J \ll J_i \), such that

\[
x_i(t, u_i) \geq x_i^*(t) - a_i + x_i(0) \quad \forall t \geq 0
\]

for all \( i \in \{1, \ldots, m\} \). Set \( u_i(t) = u_{i,\text{max}} \) for all \( i \) has no predecessor. With this input we have \( x_i(T_i) \geq t_{a_i} + a_i^* (u_i) \) (by (25)) and \( x_i(t) \geq x_i^*(t) + d \) for all \( t > 0 \) and \( J_J \gg J_i \) (since \( u \in \mathcal{U} \)).

Now let \( m + 1 \) be the first agent in topological order with initial condition \( x_{m+1} \leq a_{m+1} \). By (18) and (25) we have

\[
x_m(T_{m+1}, u_m) \geq x_{m+1}(0) + d_{m+1}^d
\]

satisfying (5)-(7), as well as (25). This reasoning can be iterated over all agents \( i \in \{m+1, \ldots, n\} \), to construct the remaining entries of the signal \( u \). The resulting \( u \in \mathcal{U}(T) \), and by (25) \( \tau_i(x_i(0)) \geq t_{a_i} + d^*_i (u_i) \) for all \( i \). Since \( \delta_{\text{max}} + T_i \geq \tau_i(x_i(0)) \) and \( P_i(T) \leq t_{a_i} + d^*_i (u_i) \), we have that \( P_i(T) \leq T_i + \delta_{\text{max}} \) for all agents \( i \) with \( x_i(0) < a_i \), as requested.

**Proof of Theorem 8**: If \( \text{APPROXIMATE SOLUTION} \) returns “no” then either \( x_i(0) \in [a_i, b_i) \) and \( x_i(0) \in [a_i, b_i) \) for some \( i, j \in I_+ \), or \( D_i < R_i \) implying \( \mathcal{U} \neq \emptyset \), or \( \text{PSEUDONOMIALTIME} \) at line 11 returns “no”. In the first case, \( x(0) \in B_+ \subset B \). In the second case, for all \( u \in U \), \( x(u, t) \) intersects \( B_\subset \subset B \). In the third case, for any \( u \in \mathcal{U} \) the schedule \( T \) defined by \( T_i = t_{a_i} (u_i) \) satisfies (16) and (20). Since \( \text{APPROXIMATE SOLUTION} \) solves \( \text{SP}^* \) exactly, and finds no feasible schedule, \( T \) must violate either (17), (18), (19). If (17) is violated then \( x_i(t, u_i) \in [a_i, b_i] \) and \( x_i(t, u_i) \in [a_i, b_i] \) for some \( t > 0 \) and for some \( i, j \in I_+ \), in which case \( x(u, t) \) intersects \( B_+ \subset \subset B \). If (18) is violated, then \( t_{a_i} (u_i) \leq t_j \) for some \( J_J \ll J_i \) with \( x_i(0) \geq a_i \) and \( x_i(0) \leq a_i \). We have that \( x_i(t_{a_i} (u_i)) \leq x_i(0) + \tau_{a_i} \delta_{\text{max}} \) and that \( t_{a_j}^d \leq \delta_{\text{max}} - (x_i(0) - a_j) / \delta_{\text{max}} \), hence \( x_i(t_{a_i} (u_i)) - x_i(t_{a_j} (u_j)) \leq \delta_{\text{max}} \delta_{\text{max}} \), that is, the trajectory intersects \( B_+ \). If (19) is violated, then \( T_i - T_j < \delta_{\text{max}} \) for some \( i, j \) with \( x_i(0) < a_i \), \( x_j(0) < a_j \), \( T_i > T_j \). This implies that \( x_i(t_{a_i} (u_i)) - x_i(t_{a_j} (u_j)) \leq \delta_{\text{max}} \delta_{\text{max}} \), that is, the trajectory intersects \( B_+ \).
Proof of Theorem 9: We prove the theorem by showing that, if \( \gamma_0 < 1 \) or \( \gamma_1 < 1 \), we can select \( \{x(0), \dot{x}(0), \Theta\} \) such that APPROXIMATESOLUTION returns "not" and construct a trajectory that does not intersect \( \bar{B} \). Consider first \( \gamma_0 < 1 \).

Take a system with two agents on two different paths. Assume

\[
\dot{x}_{1,max} = \dot{x}_{2,max} := \dot{x}_{max}, \quad \dot{x}_{1,min} = \dot{x}_{2,min} := \dot{x}_{min},
\]

\[
b_1 - a_1 = b_2 - a_2 := b - a.
\]

Let \( x_1(0) < a^1 \), \( x_2(0) < a^2 \), \( x_1(0) > a_2 \), \( \dot{x}_1(0) = \dot{x}_{max} \), and \( \dot{x}_2(0) = \dot{x}_{max} \). Select \( x_1(0) \) and \( x_2(0) \) such that \( t_{a_2}(u_{2,min}) - t_{a_1}(u_{1,max}) = \delta_{max} - \epsilon \) for some \( \epsilon > 0 \).

Now consider \( \gamma_1 < 1 \) and consider the same system as above, with identical parameters but with agents 1 and 2 on the same path. APPROXIMATESOLUTION returns "not" for the same reasons as above. Consider the trajectory \( x(t) := (x_1(t, u_{1,max}), x_2(t, u_{2,min})) \). We have that

\[
|\dot{x}_1(t, u_{1,max}) - x_2(t, u_{2,min})| \geq |x_1(0) - x_2(0)| = (\delta_{max} - \epsilon)\dot{x}_{max} - (\dot{x}_{max} - \dot{x}_{min})\delta_{a_2}(u_{2,min}).
\]

Thus, \( x(t) \notin \bar{B} \) for all \( t \geq 0 \) provided that

\[
\epsilon \in \left( 0, \delta_{max} (1 - \gamma_1) \right).
\]

ACKNOWLEDGMENT

This work was supported by the NSF Award # CNS 0930081

REFERENCES


[45] Attached Supplementary Material, will be available on authors website upon publication.