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 acceptable 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].

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: ddv@mit.edu.

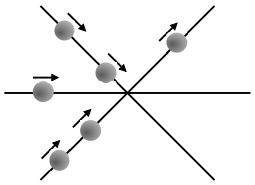

Fig. 1. Agents on 3 paths must cross an intersection while avoiding contact.

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

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:

$$\ddot{x}_i = f_i(\dot{x}_i, u_i), \tag{1}$$

with $i \in \{1, \dots, n\}$, where $x_i \in X_i \subseteq \mathbb{R}$ is the position and $u_i \in U_i \subset \mathbb{R}^m$ 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 $\mathbf{x}$ and $\mathbf{u}$ denote the vectors $(x_1, x_2, ...)$, $(u_1, u_2, ...)$, while u_i^j denotes the j-th component of vector u_i.

We call $\mathcal{U}$ the set of input signals $\mathbf{u}$, 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 $u_i \succeq u_i'$ if $u_i^j(t) \geq u_i'^j(t)$ for all j and for all $t \geq 0$, and $u_i \succ u_i'$ if $u_i^j(t) \geq u_i'^j(t)$ for all j and for all $t \geq 0$ and there exists at least one j such that $u_i^j(t) > u_i'^j(t)$ for all $t > 0$. Similarly, given a vector u_i, let $u_i \succeq u_i'$ if $u_i^j \geq u_i'^j$ for all j, and $u_i \succ u_i'$ 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 $[\dot{x}_{i,min}, \dot{x}_{i,max}]$ with nonempty interior (agents always move forward).

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

(A.5)
$$\begin{array}{c} \lim_{t\to\infty} \dot{x}_i(t, u_{i,max}) = \dot{x}_{i,max} \\ \lim_{t\to\infty} \dot{x}_i(t, u_{i,min}) = \dot{x}_{i,min} \end{array} \tag{2}$$
(min and max velocities are attained at least asymptotically by applying $u_{i,min}$ and $u_{i,max}$),

(A.6) dynamics of agents on the same path are identical, i.e., $f_i = f_j$, $[u_{i,min}, u_{i,max}] = [u_{j,min}, u_{j,max}]$, $[\dot{x}_{i,min}, \dot{x}_{i,max}] = [\dot{x}_{j,min}, \dot{x}_{j,max}]$.

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

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

while the following monotonicity property follows easily:

$$(x_i(0), \dot{x}_i(0)) \succ (x_i'(0), \dot{x}_i'(0)) \text{ and } u_i \succeq u_i' \Rightarrow x_i \succ x_i'. \tag{4}$$

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 $[\dot{x}_{i,min}, \dot{x}_{i,max}]$ 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 $i \in \{1, \dots, n\}$. 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

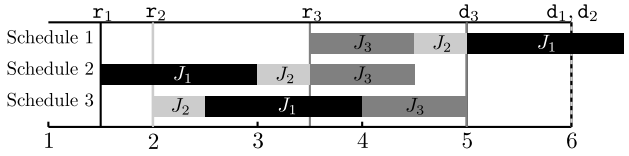

Fig. 2. Three schedules for $1|\mathtt{r_i}, \mathtt{prec}|\mathtt{Lmax}$ with $\mathtt{r} = [1.5, 2, 3.5]$, $\mathtt{d} = [6, 6, 5]$, $\mathtt{p} = [1.5, 0.5, 1]$, and $J_2 < J_1$. Schedule 1 is not feasible because J_1 terminates after its deadline. Schedule 2 is not feasible because the order of J_1 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 $B_+ := \{\mathbf{x} \in \mathbb{R}^n : \exists (i,j) \in \mathcal{I}_+, x_i \in (a_i, b_i)$ and $x_j \in (a_j, b_j)\}$ which accounts for all side impacts, and $B_- := \{\mathbf{x} \in \mathbb{R}^n : \exists (i,j) \in \mathcal{I}_-, |x_i - x_j| < d\}$, which accounts for all rear-end collisions. Given the initial condition $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$, an input signal $\mathbf{u}$ such that $\mathbf{x}(t, \mathbf{u}) \notin B$ for all $t \geq 0$ is called a *collision-free input*. VP can be formally stated as follows.

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

An instance of VP is fully described by the initial condition $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and by the tuple $\Theta := \{a_1, \ldots, a_n, b_1, \ldots, b_n, d, f_1, \ldots, f_n, X_1, \ldots, X_n, U_1, \ldots, U_n, \mathcal{U}\}$.

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

Figure 2 shows three schedules for an instance of the above problem. $1|\mathtt{r_i}, \mathtt{prec}|\mathtt{Lmax}$ was shown to be NP-Complete in [41]. It has, however, a polynomial-time exact solution, first proposed in [42], when all job durations $\mathtt{p_i}$ are identical. This case can always be transformed, by normalisation of the data, into the case where jobs have duration 1, denoted $1|\mathtt{r_i}, \mathtt{prec}, \mathtt{p_i} = 1|\mathtt{Lmax}$.

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 $I \in P1 \Leftrightarrow I' \in P2$. In this case, we write $P1 \propto P2$. If $P1 \propto P2$ and $P2 \propto P1$ then we say that $P1$ and $P2$ are equivalent, denoted $P1 \simeq P2$.

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 "$\preceq$" [43] to each set. We denote the maximum and minimum of a set in the preorder "$\preceq$" by "$\max_{\preceq}$" and "$\min_{\preceq}$", 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 $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and for fixed parameters Θ. Hence, for the sake of brevity, the dependence of all objects on $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ and Θ 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 $x_i(0) > x_j(0)$ and $(i,j) \in \mathcal{I}_-$. The set of all precedence constraints defines a directed acyclical 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 $\mathbf{T} \in \mathbb{R}^n_+$ 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 j is such that $J_j < J_i$ and $J_k < J_j$ for all $k \neq j$ such that $J_k < J_i$. In other words, the predecessor of i is the agent immediately preceding i in the topological order. This is denoted $J_j \ll J_i$. If i has no predecessor, we write $\emptyset \ll J_i$. Given an assignment of indices to agents we say that these are numbered in topological order if, for all $(i,j) \in \mathcal{I}_-$, $j < i$ implies $J_j < J_i$, in reverse topological order if $j < i$ implies $J_i < J_j$.

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

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

Theorem 1. *If $\underline{\mathcal{U}} \neq \emptyset$, then it has a minimum in the preorder "$\overset{o}{\preceq}$".*

A minimum of $\underline{\mathcal{U}}$ 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 "$\overset{o}{\preceq}$".

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

$$x_i(0) \leq a_i \Rightarrow x_i(t, u_i) \leq a_i \ \forall\ t \leq T_i, \tag{5}$$

$$J_j \ll J_i \Rightarrow x_i(t, u_i) \leq x_j(t, u_j) - d \ \forall\ t \geq 0, \tag{6}$$

$$x_i(t, u_i) \geq \underline{x}_i(t)\ \forall t \geq 0. \tag{7}$$

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 i maintain a trajectory not lower than the one obtained with the minimal input $\underline{u}_i$. Note that (7) is redundant when (6) is satisfied by all agents. It is nonetheless used in the definition of the input set $\bar{\mathcal{U}}_i(x_j, T_i)$ which follows.

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

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

The set $\bar{\mathcal{U}}(\mathbf{T})$ contains all the inputs that agents can use without violating (5) and (6). Notice that $\bar{\mathcal{U}}(\mathbf{T}) \subset \underline{\mathcal{U}}$, 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 $u_i, u_i' \in \bar{\mathcal{U}}_i(\cdot, T_i)$, we write $u_i \overset{\bar{o}}{\preceq} u_i'$ if $x_i(t, u_i) \leq x_i(t, u_i')\, \forall\, t \geq T_i$, and we say that $\mathbf{u} \overset{\bar{o}}{\preceq} \mathbf{u}'$ if the above relation holds for all $i \in \{1, \ldots, n\}$. We now address the existence of maxima of $\bar{\mathcal{U}}(\mathbf{T})$ and, to do so, we first establish the existence of maxima of $\bar{\mathcal{U}}_i(\cdot, T_i)$.

Lemma 2. *Consider agents i and j. If $J_j \ll J_i$, then take u_j such that $u_j = u_{j,max}$ for all $t \geq t_f$ for some finite $t_f \geq 0$, and consider the corresponding trajectory x_j. Assume $\underline{\mathcal{U}} \neq \emptyset$ and $\bar{\mathcal{U}}_i(\cdot, T_i) \neq \emptyset$. Then $\bar{\mathcal{U}}_i(\cdot, T_i)$ has maxima in the preorder "$\overset{\bar{o}}{\preceq}$", and for every maximum $\bar{u}_i$ there exists a finite $t_f' > 0$ such that $\bar{u}_i(t) = u_{i,max}$ for all $t \geq t_f'$. Moreover, if i has a predecessor j, then $\Big(u_j' \overset{\bar{o}}{\preceq} u_j, \quad u_i \in \max_{\overset{\bar{o}}{\preceq}} \bar{\mathcal{U}}_i(x_j(u_j), T_i), \quad u_i' \in \bar{\mathcal{U}}_i(x_j(u_j'), T_i)\Big) \Rightarrow \Big(u_i' \overset{\bar{o}}{\preceq} u_i\Big)$.*

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

$$u_{i,t_1,t_2,t_3} := \begin{cases} \underline{u}_i \text{ if } t < \min(t_1, t_3) \\ u_{i,max} \text{ if } t \in (\min(t_1, t_3), \min(t_2, t_3)] \\ u_{i,min} \text{ if } t \in (\min(t_2, t_3), t_3] \\ u_j \text{ if } t > t_3. \end{cases} \tag{8}$$

In the algorithm the function Root $F(\gamma)$ returns a positive root of the equation $F(\gamma) = 0$. If 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 $\bar{\mathcal{U}}_i(\cdot, T_i)$ 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 $\bar{\mathcal{U}}_i(\cdot, T_i)$

$\alpha \leftarrow \min(x_i(T_i, u_{i,max}), a_i)$
$\tilde{\tau}_1 \leftarrow$ Root $F(\tau_1) := x_i(T_i, u_{i,\tau_1,\infty,\infty}) - \alpha$
$\tau_2 \leftarrow$ Root $F(\tau_2) := x_i(T_i, u_{i,0,\tau_2,\infty}) - \alpha$
if i has no predecessor, or j is predecessor of i and $x_j(t, \bar{u}_j) \geq x_i(t, u_{i,\tilde{\tau}_1,\infty,\infty}) + d \forall\, t \geq 0$ **then**
  $\bar{u}_i \leftarrow u_{i,\tilde{\tau}_1,\infty,\infty}$
else ▷ assume $J_j \ll J_i$
  if $x_i(0) \leq a_i$ **then**
    $Dist_a \leftarrow \min_{t \geq T_i} \left\{x_j(t, \bar{u}_j) - x_i(t, u_{i,\tilde{\tau}_1,T_i,\infty})\right\}$
    $Dist_b \leftarrow \min_{t \geq T_i} \left\{x_j(t, \bar{u}_j) - x_i(t, u_{i,0,\mathcal{I}2,\infty})\right\}$
    if $Dist_a \geq 0$ **then**
      $\tau_1^* \leftarrow \tilde{\tau}_1$
      $\tau_2^* \leftarrow$ Root $F(\tau_2) := \min_{t \geq T_i} \left\{x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2,\infty}) - d\right\}$
      $t_t^* \leftarrow$ Root $F(t) := \Big(x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2^*,\infty}) - d\Big)$
      $\bar{u}_i \leftarrow u_{i,\tau_1^*,\tau_2^*,t_t^*}$
    else if $Dist_b \geq 0$ **then**
      $v_{T_i} \leftarrow$ Root $F(\dot{x}_i(T_i)) :=$
      $\min_{t \geq 0} \{x_j(t + T_i, \bar{u}_j) - x_i(t, u_{i,min}, a_i, \dot{x}_i(T_i)) - d\}$
      $(\tau_1^*, \tau_2^*) \leftarrow$ Root $\{F_1(\tau_1, \tau_2) := x_i(T_i, u_{i,\tau_1,\tau_2,\infty}) - \alpha,$
      $F_2(\tau_1, \tau_2) := \dot{x}_i(T_i, u_{i,\tau_1,\tau_2,\infty}) - v_{T_i}\}$
      $t_t^* \leftarrow$ Root $F(t) := \Big(x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2^*,\infty}) - d\Big)$
      $\bar{u}_i \leftarrow u_{i,\tau_1^*,\tau_2^*,t_t^*}$
    else
      $\tau_1^* \leftarrow 0$
      $\tau_2^* \leftarrow$ Root $F(\tau_2) := \min_{t \geq T_i} \left\{x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2,\infty}) - d\right\}$
      $t_t^* \leftarrow$ Root $F(t) := \Big(x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2^*,\infty}) - d\Big)$
      $\bar{u}_i \leftarrow u_{i,\tau_1^*,\tau_2^*,t_t^*}$
  else
    $\tau_1^* \leftarrow 0$
    $\tau_2^* \leftarrow$ Root $F(\tau_2) := \min_{t \geq 0} \left\{x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2,\infty}) - d\right\}$
    $t_t^* \leftarrow$ Root $F(t) := \Big(x_j(t, \bar{u}_j) - x_i(t, u_{i,\tau_1^*,\tau_2^*,\infty}) - d\Big)$
    $\bar{u}_i \leftarrow u_{i,\tau_1^*,\tau_2^*,t_t^*}$
return $\bar{u}_i$

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

$$\begin{aligned} \emptyset \ll J_i \quad &\Rightarrow \quad u_i \in \max_{\overset{\bar{o}}{\preceq}} \bar{\mathcal{U}}_i(\emptyset, T_i), \\ J_j \ll J_i \quad &\Rightarrow \quad u_i \in \max_{\overset{\bar{o}}{\preceq}} \bar{\mathcal{U}}_i(x_j(t, u_j), T_i), \end{aligned} \tag{9}$$

then $\mathbf{u} \in \max_{\overset{\bar{o}}{\preceq}} \bar{\mathcal{U}}(\mathbf{T})$.

The above result implies that the problem of finding a maximum of $\bar{\mathcal{U}}(\mathbf{T})$ 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 $\bar{\mathcal{U}}(\mathbf{T})$ is no harder than finding a maximum of $\bar{\mathcal{U}}_i(\cdot, T_i)$. Here lies the keystone of the paper: once a feasible schedule $\mathbf{T}$ has been found, finding a maximum of $\bar{\mathcal{U}}(\mathbf{T})$, and hence an input signal satisfying VP, is a simple task. The complexity of VP is entirely due to finding $\mathbf{T}$. 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 $\underline{\mathcal{U}}$ and $\bar{\mathcal{U}}(\mathbf{T})$ 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 $t_\alpha(u_i) := \min\{t \geq 0 : x_i(t, u_i) \geq \alpha\}$, that is, $t_\alpha(u_i)$ is the earliest time when x_i becomes greater than or equal to α. Such $t_\alpha(u_i)$ exists since $\dot{x}_i$ is positive and bounded away from 0. As before, let $\underline{\mathbf{u}}$ denote an element of $\min_{\preceq} \underline{\mathcal{U}}$. The scheduling quantities are defined as follows. If $\underline{\mathcal{U}} = \emptyset$, set $R_i = 0$, $D_i = -1$ for all $i \in \{1, \ldots, n\}$, otherwise

$$\begin{array}{rcl} R_i & := & t_{a_i}(u_{i,max}), \\ D_i & := & t_{a_i}(\underline{u}_i). \end{array} \tag{10}$$

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

$$P_i(\mathbf{T}) := t_{b_i}(\bar{u}_i(\mathbf{T})). \tag{11}$$

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

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

and for all $(i, j) \in \mathcal{I}_+$, if $x_i(0) < b_i$, then

$$T_i \geq T_j \Rightarrow T_i \geq P_j(\mathbf{T}). \tag{13}$$

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 $\mathbf{T}$. Notice that instances of Problems VP and SP are described by the same tuple $\{(\mathbf{x}(0), \dot{\mathbf{x}}(0)), \Theta\}$. 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 $\{(\mathbf{x}(0), \dot{\mathbf{x}}(0)), \Theta\} \in$ VP if and only if $\{(\mathbf{x}(0), \dot{\mathbf{x}}(0)), \Theta\} \in$ SP. This is done in Theorem 5. In the proof we use the following Corollary of Theorem 3.

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

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

Theorem 5. *VP $\simeq$ SP.*

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

($\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\} \in$ VP $\Rightarrow$ $\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\} \in$ SP): Assume that $\tilde{\mathbf{x}}(t, \tilde{\mathbf{u}})$ satisfies the constraints of VP. The time instants at which $\tilde{\mathbf{x}}(t, \tilde{\mathbf{u}})$ crosses each of the planes $x_i = a_i$ define a vector $\mathbf{T}$, and we can set $T_i = 0$ if $x_i(0) > a_i$. Given this choice of $\mathbf{T}$, $\bar{\mathcal{U}}(\mathbf{T}) \neq \emptyset$, and since $\bar{\mathcal{U}}(\mathbf{T}) \subseteq \underline{\mathcal{U}}$, $\underline{\mathcal{U}} \neq \emptyset$, so by Theorem 1 the quantities $\underline{x}_i(t)$ exist. Moreover this $\mathbf{T}$ satisfies (12) given the definition of R_i and D_i. The time instants at which $\tilde{\mathbf{x}}(t, \tilde{\mathbf{u}})$ crosses the planes $x_i = b_i$ define a vector $\tilde{\mathbf{P}} = (\tilde{P}_1 \ldots, \tilde{P}_n)$, with $\tilde{P}_i = t_b(\tilde{u}_i)$, and for all i such that $T_i \geq T_j$, $(i, j) \in \mathcal{I}_+$ and $x_i(0) < b_i$, we have $T_i \geq \tilde{P}_j$, otherwise $\tilde{\mathbf{x}}(t, \tilde{\mathbf{u}}) \cap B \neq \emptyset$. By Corollary 4, $P_i(\mathbf{T}) \leq t_b(\tilde{u}_i) = \tilde{P}_i$ for all $\mathbf{u} \in \bar{\mathcal{U}}(\mathbf{T})$, so we conclude that $\mathbf{T}$ satisfies (13).

($\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\} \in$ VP $\Leftarrow$ $\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\} \in$ SP): Assume that $\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\} \in$ SP, i.e., there exists a schedule $\mathbf{T}$ that satisfies SP given $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$. In order for any schedule $\mathbf{T}$ to satisfy (12), it must be that $\underline{\mathcal{U}} \neq \emptyset$. If for all $(i, j) \in \mathcal{I}_+$, $x_i(0) \geq b_i$ and $x_j(0) \geq b_j$, then any input in $\underline{\mathcal{U}}$ solves VP. Otherwise, in order to satisfy (13) the values of $P_i(\mathbf{T})$ must be finite. By the definition of $P_i(\mathbf{T})$, this implies that $\bar{\mathcal{U}}(\mathbf{T}) \neq \emptyset$ and that $t_{b_i}(\bar{u}_i) = P_i(\mathbf{T})$, where $\bar{\mathbf{u}} \in \max_{\bar{o}} \bar{\mathcal{U}}(\mathbf{T})$. This, together with the fact that all inputs in $\bar{\mathcal{U}}(\mathbf{T})$ satisfy (5) and (6) and that $\dot{x}_i(t) > \dot{x}_{i,min} > 0$ for all t, insures that: (i) if $x_i(0) \leq a_i$, then $x_i(t, \bar{u}_i) \leq a_i$ for all $t < T_i$, (ii) if $x_i(0) < b_i$ then $x_i(t, \bar{u}_i) < b_i$ for all $t < P_i(\mathbf{T})$, and (iii) for all $t \geq 0$, $x_i(t, \bar{u}_i) \leq x_j(t, \bar{u}_j) - d$ 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 $P_i(\mathbf{T})$, and the intervals $(T_i, P_i(\mathbf{T}))$ and $(T_j, P_j(\mathbf{T}))$ with $(i, j) \in \mathcal{I}_+$ do not intersect. Thus $\mathbf{x}(\bar{\mathbf{u}}, t) \notin B$ for all $t \geq 0$, and $\bar{\mathbf{u}}$ satisfies VP. ■

B. Exact Solution of VP

By Theorem 5 a solution of VP, which is defined over a function space, can by 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 $\mathbf{T}$ with elements T_i in topological order, that is, such that $J_j \ll J_i \Rightarrow T_j \leq T_i$.*

Proof: The property follows from the fact that, if $J_j \ll J_i$, 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 $T_i \geq T_j \Rightarrow T_i = P_j(\mathbf{T})$. Such a schedule is uniquely determined once the order of jobs has been chosen. Therefore, the search space coincides with the set $\mathcal{P}$ of all possible permutations of the indices $1, \ldots, n$ that satisfy the topological order ($J_j < J_i \Rightarrow j < i$). The procedure EXACTSOLUTION in Algorithm 2 solves SP exactly by performing an exhaustive search in $\mathcal{P}$. We denote by $\pi \in \mathcal{P}$ a permutation of indices and by π_i the i-th index in the permutation. With some abuse of notation, in the procedure we write $\tilde{P}_i \leftarrow t_{b_i}\Big(\max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}_i(\cdot, T_i)\Big)$ as a short form of $u_i \in \max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}_i(\cdot, T_i)$, $\tilde{P}_i \leftarrow t_{b_i}(u_i)$. The abbreviation should cause no confusion, since as we noted before, all maxima of $\max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}(\cdot, T_i)$ give the same trajectory for $t \geq T_i$, and therefore the same value of t_{b_i}. The state $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ is in

Algorithm 2 Exact solution of SP

1: **procedure** EXACTSOLUTION($\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta$)
2: **for all** $i \in \{1, \ldots, n\}$ **do**
3: given $x_i(0)$ and Θ calculate R_i, D_i
4: **for all** $\pi \in \mathcal{P}$ **do**
5: $T_{\pi_1} \leftarrow R_{\pi_1}$
6: $\bar{u}_{\pi_1} \leftarrow \max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}_1(\emptyset, T_{\pi_1})$
7: $\tilde{P}_{\pi_1} \leftarrow t_{b_{\pi_1}}(\bar{u}_{\pi_1})$
8: **for** $i = 2 \rightarrow n$ **do**
9: **if** $(\pi_i, \pi_{i-1}) \in \mathcal{I}_+$ **then**
10: $T_{\pi_i} \leftarrow \max\{\tilde{P}_{\pi_{i-1}}, R_{\pi_i}\}$
11: **else**
12: $T_{\pi_i} \leftarrow \max\{T_{\pi_{i-1}}, R_{\pi_i}\}$
13: **if** $\exists j : J_j \ll J_{\pi_i}$ **then**
14: **if** $\bar{\mathcal{U}}_{\pi_i}(x_{\pi_{i-1}}, T_{\pi_i}) \neq \emptyset$ **then**
15: $\bar{u}_{\pi_i} \leftarrow \max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}_i(x_j(\bar{u}_j), T_i)$
16: $\tilde{P}_{\pi_i} \leftarrow t_{b_{\pi_i}}(\bar{u}_{\pi_i})$
17: **else**
18: $\tilde{P}_{\pi_i} \leftarrow \infty$
19: **else**
20: **if** $\bar{\mathcal{U}}_{\pi_i}(\emptyset, T_{\pi_i}) \neq \emptyset$ **then**
21: $\bar{u}_{\pi_i} \leftarrow \max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}_i(x_j(\emptyset, T_i))$
22: $\tilde{P}_{\pi_i} \leftarrow t_{b_{\pi_i}}(\bar{u}_{\pi_i})$
23: **else**
24: $\tilde{P}_{\pi_i} \leftarrow \infty$
25: **if** $T_i \leq D_i \, \forall \, i \in \{1, \ldots n\}$ **then**
26: **return** $\{\mathbf{T}, yes\}$
27: **return** $\{\emptyset, no\}$

the maximal controlled invariant set if and only if EXACTSOLUTION finds a feasible schedule. If a feasible schedule is found, the input $\mathbf{u} \in \max\nolimits_{\bar{\preceq}} \bar{\mathcal{U}}(\mathbf{T})$ constructed at lines 6 and 21 of Algorithm 2 is a safe input for $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$.

Example 1 (Execution of Algorithm 2)**.** Consider three identical agents on two paths. Model the agents' longitudinal dynamics as a linear double integrator with saturation, so that the vector field in (1) takes the form

$$f_i(\dot{x}_i, u_i) = \begin{cases} u_i \text{ if } (\dot{x}_i < \dot{x}_{i,max} \text{ and } u_i > 0) \text{ or} \\ \quad (\dot{x}_i > \dot{x}_{i,min} \text{ and } u_i < 0) \\ 0 \text{ otherwise} , \end{cases}$$

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 $\dot{x}_i(0) = 1$ for all agents, and let agents 1 and 2 be on the same path. The initial conditions imply that $J_2 \ll J_1$.

We have that $\underline{u}_i(t) = -1$ 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 $\underline{u}_i$ computed above are, respectively, $R_i \simeq \{2.317, 2, 2.317\}$ and $D_i = \{5, 4, 5\}$. Lines 5-7 of Algorithm 2 assign $T_2 = 2$, $\bar{u}_2 = 1$, $\tilde{P}_2 \simeq 2.317$. 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 $\tilde{\tau}_1 = 0$ and $x_2(t, 1) \geq x_1(t, u_{1,0,\infty})$, so that the assignment at Line 5 of Algorithm 1 sets $\bar{u}_1 = 1$ for all $t \geq 0$. This gives $\tilde{P}_1 \simeq 2.606$. Finally, Lines 12, 21, and 22 of Algorithm 2 assign $T_3 \simeq 2.606$, $\bar{u}_3(t) = 0$ for $t \in [0, 0.418]$, $\bar{u}_3(t) = u_{3,max}$ for $t > 0.418$, and $\tilde{P}_3 \simeq 2.488$. We thus obtain $\mathbf{T} = [2, 2, 2.606]$, 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 seek an approximate solution.

C. Approximate 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 $1|\mathtt{r_i}, \mathtt{prec}, \mathtt{p_i} = 1|\mathtt{Lmax}$ (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 $1|\mathtt{r_i}, \mathtt{prec}, \mathtt{p_i} = 1|\mathtt{Lmax}$. The idea is to define a time δ_{max} long enough so that any agent is able to cross the interval $[a_i, b_i]$ in at most δ_{max}, and allocate this fixed amount of time to each agent.

To begin with, consider the quantity

$$\begin{aligned} d_i^* := \inf\{\alpha : \exists (u_i, u_j), x_i(u_i) + d \preceq x_j(u_j), \\ x_i(0) + \alpha = x_j(0), \dot{x}_i(0) = \dot{x}_{i,max}, \dot{x}_j(0) = \dot{x}_{j,min}, \\ (i, j) \in \mathcal{I}_-\} \end{aligned}$$

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

$$\begin{aligned} \tau_i(\alpha) := \inf\{t \geq 0 : x_i(t, u_{i,max}) \geq \max\{b_i, a_i + d_i^*\}, \\ x_i(0) = \alpha, \dot{x}_i(0) = \dot{x}_{i,min}\}. \end{aligned} \tag{14}$$

This is the minimum time taken to reach $\max\{b_i, a_i + d_i^*\}$ by an agent that starts at position α with minimum velocity, using input $u_{i,max}$. Finally, call

$$\delta_{max} := \max_{i\in\{1,\dots,n\}} \tau_i(a_i). \tag{15}$$

We can now define the approximation SP* of SP:

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

$$R_i \le T_i \le D_i, \tag{16}$$

for all $(i,j) \in \mathcal{I}_+$, if $T_j = 0$ and $x_i(0) < b_i$ then

$$T_i \ge T_j \Rightarrow T_i \ge P_j(\mathbf{T}), \tag{17}$$

for all $(i,j) \in \mathcal{I}_-$ if $T_j = 0$ then

$$J_j < J_i \Rightarrow T_i \ge \tau_j(x_j(0)) \tag{18}$$

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

$$T_i \ge T_j \Rightarrow T_i \ge T_j + \delta_{max} \tag{19}$$

and

$$J_j < J_i \Rightarrow T_i \ge T_j. \tag{20}$$

Note that, by (16), $T_j = 0$ if and only if $x_j(0) \ge a_j$. Constraints (16) and (17) are the same as in 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 $a_j + d_j^*$. 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 δ_{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 $x_i(0) \ge a_i$ for $i = 1, \dots, m$, and $x_i(0) < a_i$ for $i = m+1, \dots, n$. $P_j(0)$ at Line 7 stands for $P_j(\mathbf{T})$ with $T_i = 0$ for all i.

Algorithm 3 Solution of SP*

1: **procedure** APPROXIMATESOLUTION($\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta$)
2: **for all** $i \in \{1, \dots, n\}$ **do** given $x_i(0)$ calculate R_i, D_i
3: **if** $x_i(0) \in [a_i, b_i)$ and $x_j(0) \in [a_j, b_j)$ for some $(i,j) \in \mathcal{I}_+$ or $D_i < R_i$ for some i **then**
4: **return** $\{\emptyset$, no$\}$
5: **for all** $i \in \{1, \dots, m\}$ **do** $T_i \leftarrow 0$
6: **for all** $i \in \{m+1, \dots, n\}$ **do**
7: $\tilde{R}_i \leftarrow \max\left\{ \max_{j\le m:(i,j)\in\mathcal{I}_+} \{P_j(0)\}, \max_{j\le m:(i,j)\in\mathcal{I}_-} \{\tau_j\}, R_i \right\}$
8: set δ_{max} as in (15)
9: $\mathbf{r} = (\tilde{R}_{m+1}/\delta_{max}, \dots, \tilde{R}_n/\delta_{max})$
10: $\mathbf{d} = (D_{m+1}/\delta_{max} + 1, \dots, D_n/\delta_{max} + 1)$
11: $\{\mathtt{T}_{m+1}, \dots, \mathtt{T}_n$, answer$\}$= POLYNOMIALTIME($\mathbf{r}, \mathbf{d}$, prec)
12: **for** $i = m+1 \to n$ **do** $T_i \leftarrow \mathtt{T}_i \delta_{max}$
13: **return** $\{\mathbf{T}$, 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 $a_i = 30$, $b_i = 31$, $d = 1$, $\dot{x}_i(0) = 1$ for all agents. We can compute d_i^* and δ_{max} explicitly, obtaining $d_i^* = 21.25$ and $\delta_{max} \simeq 6.37$. All agents have $x_i(0) < a_i$, therefore $\tilde{R}_i = R_i$ in Algorithm 3. By proceeding as in Example 1 we obtain $R = \tilde{R} \simeq [7.62, 7.07, 7.62]$ and $D = [30, 26, 30]$. The vectors $\mathtt{r}$ and $\mathtt{d}$ obtained by dividing $\tilde{R}$ and D by δ_{max} are $\mathtt{r} \simeq [2.11, 1.11, 1.19]$ and $\mathtt{d} \simeq [4.71, 4.08, 4.71]$. The procedure POLINOMIALTIME finds the feasible schedule $\mathtt{T} \simeq [3.11, 1.11, 2.11]$, which corresponds to $T \simeq [19.8, 7.07, 13.43]$.

We say that SP* is an approximation of SP if any schedule that is feasible for SP* is feasible also for SP.

Theorem 7. *SP* is an approximation of SP.*

By this theorem, APPROXIMATESOLUTION 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 δ_{max}. The following theorem provides a measure of the extent of the underestimation. First, define the sets $\hat{B}_+ := \{\mathbf{x} : \exists (i,j) \in \mathcal{I}_+, x_i \in [a_i, a_i + \delta_{max}\dot{x}_{max}], x_j \in [a_j, a_j + \delta_{max}\dot{x}_{max}]\}$ and $\hat{B}_- := \{\mathbf{x} : \exists (i,j) \in \mathcal{I}_-, |x_i - x_j| \le \delta_{max}\dot{x}_{max}\}$ and define the *extended bad set*

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

The extended bad set is thus a superset of the bad set. In Theorem 8 we prove that if APPROXIMATESOLUTION returns "no" then, for all $\mathbf{u} \in \mathcal{U}$, 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* $(\mathbf{x}(0), \dot{\mathbf{x}}(0))$ APPROXIMATESOLUTION *returns "no", then for all* $\mathbf{u} \in \mathcal{U}$ *there exists a* $t \ge 0$ *such that* $\mathbf{x}(t, \mathbf{u}) \in \hat{B}$.

Now define the sets $\check{B}_+ := \{\mathbf{x} : \exists (i,j) \in \mathcal{I}_+, x_i \in [a_i, a_i + \gamma_+\delta_{max}\dot{x}_{max}], x_j \in [a_j, a_j + \gamma_+\delta_{max}\dot{x}_{max}]\}$ and $\check{B}- := \{\mathbf{x} : \exists (i,j) \in \mathcal{I}_-, |x_i - x_j| \le \gamma_-\delta_{max}\dot{x}_{max}\}$ and let $\check{B} := \check{B}_+ \cup \check{B}_-$.

Theorem 9. *If* $\gamma_+ < 1$ *or* $\gamma_- < 1$*, there exists a tuple* $\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\}$ *such that* APPROXIMATESOLUTION *returns "no" and, for at least one* $\mathbf{u} \in \mathcal{U}$, $\mathbf{x}(t, \mathbf{u}) \notin \check{B}$ *for all* $t \ge 0$.

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 τ, we design the supervisor as a map $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k) \mapsto \mathbf{u}_k$ for all $k \in \mathbb{N}$. The map takes a "desired" input $\mathbf{v}_k \in U := (U_1, \dots, U_n)$ for the time interval $[k\tau, (k+1)\tau)$, and returns a signal $\mathbf{u}_k(t) = \mathbf{v}_k$ 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 $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ is explicitly included among the arguments of the trajectories. Let $\mathbf{v}_k$ denote the desired input at time $k\tau$. Consider the two signals $\mathbf{u}_k$ and $\mathbf{u}_k^\infty$ defined as follows: the first one is defined on the interval $[k\tau, (k+1)\tau]$ and identically equal to $\mathbf{v}_k$; the second one is an element of $\mathcal{U}$ defined on $[k\tau, \infty)$, and such that $\mathbf{u}_k^\infty(t) = \mathbf{u}_k(t)$ when $t \in [k\tau, (k+1)\tau]$. Additionally, given $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$, call $\mathbf{u}_{k,safe}^\infty \in \mathcal{U}$ a control signal

such that $\mathbf{x}(t, \mathbf{u}^{\infty}_{k,safe}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)) \notin B$ for all $t \geq k\tau$ (if such control exists), and call $\mathbf{u}_{k,safe}$ the restriction of $\mathbf{u}^{\infty}_{k,safe}$ to the interval $[k\tau, (k+1)\tau]$. If $\mathbf{u}^{\infty}_{k,safe}$ does not exist, let $\mathbf{u}^{\infty}_{k,safe}, \mathbf{u}_{k,safe} = \emptyset$. The supervisor design problem is formally stated as follows

Problem 1. Design a supervisor $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ such that

$$s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k) = \begin{cases} \mathbf{u}_k & \text{if } \exists\, \mathbf{u}^{\infty}_k(t) \in \mathcal{U} : \\ & \mathbf{x}(t, \mathbf{u}^{\infty}_k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau)) \notin B \\ & \quad \forall\ t \geq 0 \\ \mathbf{u}_{k,safe} & \text{otherwise.} \end{cases}$$

The above supervisor overrides the desired input $\mathbf{v}_k$ whenever this will cause a collision at some future time. Moreover, it has the following property.

Proposition 10. *The supervisor $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ defined above is nonblocking, i.e., if $\mathbf{u}_k := s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k) \neq \emptyset$ and $\mathbf{x}_{k+1} = \mathbf{x}(\tau, \mathbf{u}_k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$, $\dot{\mathbf{x}}_{k+1} = \dot{\mathbf{x}}(\tau, \mathbf{u}_k, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$, then for any $\mathbf{v}_{k+1}$, $s(\mathbf{x}_{k+1}, \dot{\mathbf{x}}_{k+1}, \mathbf{v}_{k+1}) \neq \emptyset$.*

Sketch of proof: Nonblockingness follows from the associative property of the flow action: if at time $k\tau$ 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 $(k+1)\tau$. ■

This ensures that at all times $k\tau$, and for all desired inputs $\mathbf{v}_k$, the supervisor returns a collision-free input.

Given a system of the form (1) and the state $(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$ at some time $k\tau$, the procedure EXACTSOLUTION returns a binary value (yes/no), and a schedule $\mathbf{T}$. We can use this information to design the supervisor in Problem 1. Assume that, at $t = 0$, we have EXACTSOLUTION$(\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta) = \{\mathbf{T}_0, yes\}$. Define $\mathbf{u}^{\infty}_{0,safe} = \bar{\mathbf{u}}(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{T}_0)$, where $\bar{\mathbf{u}}(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{T}_0) \in \bar{\mathcal{U}}(\mathbf{T}_0)$, and we have explicitly written the initial conditions for clarity. Define $\mathbf{u}_{0,safe}$ as the restriction of $\mathbf{u}^{\infty}_{0,safe}$ to the time interval $[0, \tau]$. At each iteration $k = 0, 1, 2, \ldots$, the supervisor map $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ is defined in Algorithm 4.

Algorithm 4 Implementation of the supervisor map

```
1:  procedure s(x(kτ), ẋ(kτ), v_k)
2:      ū(t) ← v_k ∀ t ∈ [kτ, (k+1)τ]
3:      x_{k+1} ← x((k+1)τ, ū, x(kτ), ẋ(kτ))
4:      ẋ_{k+1} ← ẋ((k+1)τ, ū, x(kτ), ẋ(kτ))
5:      {T, answer} ← ExactSolution(x_{k+1}, ẋ_{k+1}, Θ)
6:      if (answer = yes) and x(t, ū) ∉ B for all t ∈ [kτ, (k+1)τ]
    then
7:          return ū
8:      else
9:          {T, answer} ← ExactSolution(x(kτ), ẋ(kτ), Θ)
10:         u^∞_{k,safe} ← ū(x(kτ), ẋ(kτ), T)
11:         u_{k,safe} ← u^∞_{k,safe} restricted to [kτ, (k+1)τ]
12:         return u_{k,safe}
```

Theorem 11. *Assume that $s(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{v}_0) \neq \emptyset$. Then, the supervisor defined by Algorithm 4 solves Problem 1.*

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

Algorithm 4 uses EXACTSOLUTION, whose running time scales multinomially 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 EXACTSOLUTION with APPROXIMATESOLUTION at lines 5 and 9 of the Algorithm. Through this substitution we obtain a supervisor, denoted $s_{approx}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$, 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 $\hat{B}$ defined in (21). Call $\hat{s}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ the supervisor defined in Problem 1 substituting $\hat{B}$ to B.

Theorem 12. *$s_{approx}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ is no more restrictive than $\hat{s}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$, that is, if $s_{approx}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k) = \mathbf{u}_{k,safe}$ then $\hat{s}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k) = \mathbf{u}_{k,safe}$. Moreover if $s_{approx}(\mathbf{x}(0), \dot{\mathbf{x}}(0), \mathbf{v}_0) \neq \emptyset$ then s_{approx} is nonblocking in the sense defined in Proposition 10.*

Sketch of proof: The fact that $s_{approx}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ is no more restrictive than $\hat{s}(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$ 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 $\mathbf{u}_{k-1,safe}$. Given the trajectory $\mathbf{x}(t, \mathbf{u}^{\infty}_{k-1,safe})$, let $\mathbf{T}_{k-1}$ be the vector of times at which agents starting at $\mathbf{x}((k-1)\tau)$ enter the intersection, with $T_{k-1,i} = 0$ if $x_i(k-1\tau) \geq a_i$. Let $\mathbf{T}_k$ be the same vector computed from $\mathbf{x}(k\tau, \mathbf{u}^{\infty}_{k,safe})$. On can check that if $\mathbf{T}_{k-1}$ satisfies (16)-(20) then so does $\mathbf{T}_k$, 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 supervisory algorithms described in Section V on a set of vehicles governed by the equation

$$\ddot{x}_i = \begin{cases} u_i - 0.005(\dot{x}_i)^2 \text{ if} \\ \quad (\dot{x}_i > \dot{x}_{i,min} \text{ and } u - 0.005(\dot{x}_i)^2 \leq 0) \text{ or} \\ \quad (\dot{x}_i < \dot{x}_{i,max} \text{ and } u - 0.005(\dot{x}_i)^2 \geq 0) \\ 0 \text{ otherwise,} \end{cases} \tag{22}$$

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: $b_i - a_i = 10$m, $\dot{x}_{i,min} = 1.39$m/s, $\dot{x}_{i,max} = 13.9$m/s, $u_{i,min} = -2$m/s^2, $u_{i,max} = 2$m/s^2, $d = 5$m. This gives $b^*_i \simeq a_i + 16.998$m and $\delta_{max} \simeq 3.6686$s.

Algorithm 4 was implemented numerically, as explained in [45], with $\tau = 0.2$s. 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

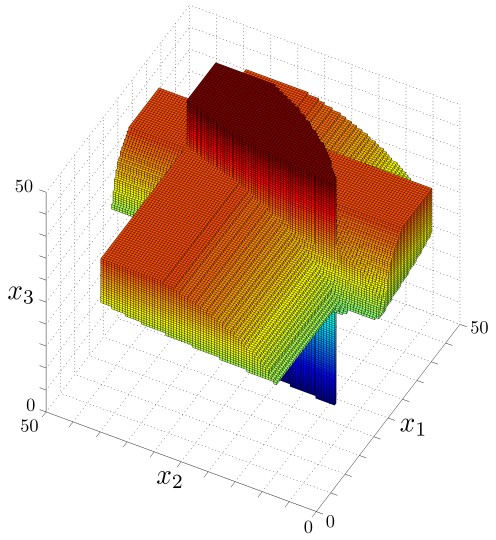

Fig. 3. The capture set (the complement of the maximal controlled invariant set) for 3 agents on 3 different paths. Initial velocities are $[5, 12, 3]$.

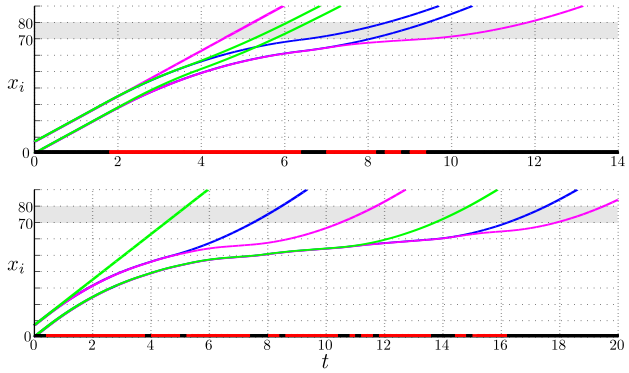

Fig. 4. Supervisor of 6 agents on 3 paths, $\dot{x}_i(0) = 13.9$ for all agents. The gray band marks the interval $[a_i, b_i]$, identical for all agents. Trajectories of agents on the same path are plotted in the same colour. Red segments on the t axis denote intervals where the supervisor is overriding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

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.

VII. CONCLUSION

We have proved that checking membership in the maximal controlled invariant set for n dynamic agents at the intersection

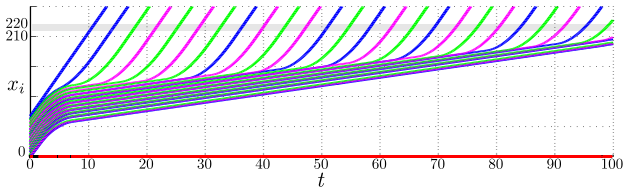

Fig. 5. Supervisor with algorithm APPROXIMATESOLUTION, 30 agents on 3 paths, $\dot{x}_i(0) = 13.9$ for all agents, colour coding as in Fig. 4.

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 $\mathbf{x}$ in the state space belongs to the maximal controlled invariant set, that is, if there exists a conflict-free trajectory starting at $\mathbf{x}$. 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 $u_i', u_i'' \in \mathcal{U}_i$, and let $u_i'(t) := u_{max}$ for all $t \in [t_1, t_2]$. Then $\big(x_i(t_1, u_i'') \geq x_i(t_1, u_i'), x_i(t_2, u_i'') \geq x_i(t_2, u_i')\big) \Leftrightarrow \big(x_i(t, u_i'') \geq x_i(t, u_i')\, \forall\, t \in [t_1, t_2]\big)$.*

Proposition 14. *Let $u_i', u_i'' \in \mathcal{U}_i$, and let $u_i'(t) := u_{min}$ for all $t \in [t_1, t_2]$. Then $\big(x_i(t_1, u_i'') \leq x_i(t_1, u_i'), x_i(t_2, u_i'') \leq x_i(t_2, u_i')\big) \Leftrightarrow \big(x_i(t, u_i'') \leq x_i(t, u_i')\, \forall\, t \in [t_1, t_2]\big)$.*

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

Proposition 15. *Let $u_{i,\alpha} \in \mathcal{U}_i, u_{j,\beta} \in \mathcal{U}_j$ 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 $u_{i,\alpha}(t) = u_{max}$ and $u_{j,\beta}(t) = u_{min}$ for all $t \geq t_s$ and for all $\alpha \in A, \beta \in B$.*

Assume that $x_i(t, u_{i,\alpha})$ and $x_j(t, u_{j,\beta})$ 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 $x_i(t, u_{i,\alpha'}) \geq x_j(t, u_{j,\beta'}) + \gamma$ for all $t \geq 0$, and a pair $(\alpha'', \beta'') \in A \times B$ such that $x_i(t, u_{i,\alpha''}) < x_j(t, u_{j,\beta''})$ for some finite t, then there exists a pair $(\alpha''', \beta''') \in A \times B$ such that $x_i(t, u_{i,\alpha'''}) \geq x_j(t, u_{j,\beta'''}) + \gamma$ for all $t \geq 0$ and $x_i(t, u_{i,\alpha'''})$ is tangent to $x_j(t, u_{j,\beta'''}) + \gamma$ at some finite t_t.

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

The following continuity property of the solutions of (1) is a consequence of (A.2) and (A.4). Consider a set $\mathbf{u}_0, \ldots, \mathbf{u}_n$ of continuous signals in $\mathcal{U}$. Let $\tau_0 = 0$, let $\tau_0 < \tau_1 < \ldots < \tau_r$ be a finite sequence of real numbers, and let $\mathbf{u}_{\tau_0,\ldots,\tau_r}$ be an input signal equal to $\mathbf{u}_k$, $k \in \{0, \ldots, r-1\}$, for $t \in [\tau_k, \tau_{k+1})$, and equal to $\mathbf{u}_r$ for $t \geq \tau_r$.

Lemma 16. *For every $\epsilon > 0$ there exists a $\delta > 0$ such that, if $|\tau'_k - \tau_k| < \delta \,\forall\, k$, then $\|\mathbf{x}(t, \mathbf{u}_{\tau'_0,\ldots,\tau'_n}) - \mathbf{x}(t, \mathbf{u}_{\tau_0,\ldots,\tau_r})\|_\infty < \epsilon \,\forall\, t \in [0, \tau_r]$.*

We can now prove Theorem 1.

Proof of Theorem 1: The proof is constructive. Since the inequality conditions defining B_- only involve pairs of agents on the same path, the set $\underline{\mathcal{U}}$ is the cross product of a number of sets equal to the number of paths, each set containing trajectories that do not intersect the restriction of B_- to the corresponding subspace. Thus, we can prove the theorem by dealing with one path at a time. Consider all agents along a path in reverse topological order, indexed from 1 to n where n is the number of agents on the path. By (3), and since U_1 is bounded, $\underline{u}_1(t) := u_{min}$ for all $t \geq 0$ is the unique input minimizing $x_1(t, u_1)$ for all $t \geq 0$. Therefore, it is the first component of all vectors in $\min_{\underset{\preceq}{o}} \underline{\mathcal{U}}$. For all following agents, we construct the components of $\min_{\underset{\preceq}{o}} \underline{\mathcal{U}}$ inductively, using the above as the base case. Assume the component $i-1$ of $\min_{\underset{\preceq}{o}} \underline{\mathcal{U}}$ is known, and call it $\underline{u}_{i-1}$. Assume also that there exists some $t_f > 0$ such that $\underline{u}_{i-1}(t) = u_{min}$ for all $t > t_f$. Let $\underline{x}_{i-1}(t) := x_{i-1}(t, \underline{u}_{i-1})$. Consider the family $u_{i,\tau}$ of inputs with elements $u_{i,\tau}(t) := \{u_{min}$ if $t \leq \tau, u_{max}$ if $t > \tau\}$ for all $\tau \in \mathbb{R}_+$. $u_{i,\tau}$ is a totally ordered family of inputs according to the above order. By (3), inputs in $u_{i,\tau}$ generate a totally ordered set of position trajectories which, by Lemma 16, depend continuously on τ. If, for all $u_{i,\tau}$, $x_i(t, u_{i,\tau}) \geq \underline{x}_{i-1} + d$, then $u_i(t) := u_{min}$ for all $t \geq 0$ is the unique input minimizing $x_i(t, u_i)$ for all $t \geq 0$ while satisfying $x_i(t, u_i) \geq \underline{x}_{i-1}(t) + d$. Therefore, it is the component i of $\min_{\underset{\preceq}{o}} \underline{\mathcal{U}}$, and $\underline{u}_i(t) = u_{min}$ for all $t \geq 0$. Otherwise, since $\underline{\mathcal{U}} \neq \emptyset$, it must be that $x_i(t, u_{i,0}) \geq x_{i-1}(t, \underline{u}_{i-1}) + d$ for all $t \geq 0$, while $x_i(t, u_{i,\tau}) < x_{i-1}(t, \underline{u}_{i-1})$ for some $t \geq 0$, for τ sufficiently large. Therefore we can use Proposition 15, with $u_{i,\alpha} = u_{i,\tau}$, $u_{i,\beta} = \underline{u}_{i-1}$, and $t_0 = 0$ to show that there exists a τ_{opt} such that $x_i(t, u_{i,\tau_{opt}}) \geq x_{i-1}(t, \underline{u}_{i-1}) + d$ for all $t \geq 0$ and $x_i(t, u_{i,\tau_{opt}})$ is tangent to $x_{i-1}(t, \underline{u}_{i-1}) + d$ at some $t_t \geq 0$. Let us set $\underline{u}_i(t) := u_{i,\tau_{opt}}$ for all $t \leq t_t$, and $\underline{u}_i(t) := \underline{u}_{i-1}(t)$ for $t > t_t$. Tangency of the trajectories implies that, at $t = t_t$, the states of agents i and j are identical except for a translation by d in the position. By (A.6) choosing identical inputs for all $t \geq t_t$ ensures that the distance d is preserved, i.e., $x_i(t, \underline{u}_i(t)) = \underline{x}_{i-1}(t) + d$ for all $t \geq t_t$. If $t_t < \tau_{opt}$, $x_i(t, \underline{u}_i) \leq x_i(t, u_i)$ for all $u_i \in \underline{\mathcal{U}}$. This is ensured in the interval $[0, t_t]$ by (3) (since we are using $u_i = u_{min}$ for all $t \leq t_t < \tau_{opt}$), and in the interval $[t_t, \infty)$ by the constraint $\mathbf{x}(t) \cap B_- = \emptyset$ (since any $x_i(t)$ below $x_i(t, \underline{u}_i)$ would lie below $\underline{x}_{i-1}(t) + d$). If instead $t_t \geq \tau_{opt}$, by Proposition 13 with $t_1 = \tau_{opt}$, $t_2 = t_t$, we have $x_i(t, \underline{u}_i) \leq x_i(t, u_i)$ for all $u_i \in \underline{\mathcal{U}}$ and for all $t \in [\tau_{opt}, t_t]$. The inequalities in the rest of the real line are proved as before.

Thus, $\underline{u}_i(t)$ is the component i of $\min_{\underset{\preceq}{o}} \underline{\mathcal{U}}$, and since $\underline{u}_i(t) := \underline{u}_{i-1}(t)$ for $t \geq t_t$, $\underline{u}_i(t) = u_{min}$ for all $t \geq t_f$, for some finite $t_f \geq 0$, completing the induction step. ■

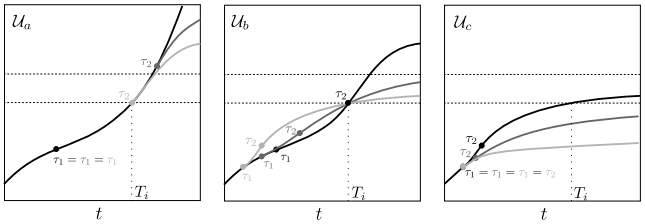

Fig. 6. The set $\mathcal{U}_{a+b+c}$ is the union of three sets of signals giving the trajectories in the panels. The dashed horizontal band is the intersection, different curves, and the corresponding parameters τ_1 and τ_2, are represented in different shades of gray.

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

(i) $\exists \bar{u}_i \in \bar{\mathcal{U}}_i(\cdot, T_i) : u_i \overset{\bar{o}}{\preceq} \bar{u}_i \,\forall\, u_i \in \bar{\mathcal{U}}_i(\cdot, T_i)$,

(ii) $\Big(J_j \ll J_i,\, u'_j \overset{\bar{o}}{\preceq} u_j, u_i \in \max_{\underset{\preceq}{\bar{o}}} \bar{\mathcal{U}}_i(x_j(u_j), T_i),\, u'_i \in \bar{\mathcal{U}}_i(x_j(u'_j), T_i)\Big) \Rightarrow u'_i \overset{\bar{o}}{\preceq} u_i$.

(iii) $\bar{u}_i(t) = u_{i,max}$ for all $t \geq t'_f > 0$.

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

$$u_i := \begin{cases} \underline{u}_i \,\forall\, t \in [0, \tau_1], \\ u_{i,max} \,\forall\, t \in (\tau_1, \tau_2], \\ u_{i,min} \,\forall\, t > \tau_2 \end{cases} \tag{23}$$

with $\tau_1 \in [0, \tau_2]$ and $\tau_2 \in [0, \infty]$. $\mathcal{U}_a$ is such that $\tau_1 \leq T_i \leq \tau_2$ and the corresponding trajectories reach a_i exactly at T_i. $\mathcal{U}_b$ is such that $\tau_1 \leq \tau_2 \leq T_i$ and the corresponding trajectories reach a_i exactly at T_i. $\mathcal{U}_c$ is such that $\tau_1 \leq \tau_2 \leq T_i$ and $x_i(T_i, u_i) \leq a_i$. It can be proven that the above family satisfies:

(p.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 "$\overset{\bar{o}}{\preceq}$",

(p.2) $\min_{\underset{\preceq}{\bar{o}}} \mathcal{U}_{a+b+c} \overset{\bar{o}}{\preceq} \underline{u}_i$,

(p.3) the values of τ_1, τ_2 representing inputs in $\mathcal{U}_{a+b+c}$ form a path connected set,

(p.4) for all $u_i' \in \mathcal{U}_{a+b+c}$ and for all $u_i'' \in \mathcal{U}$ that satisfy (5), $\big(\exists\, t_c \geq T_i : x_i(t_c, u_i'') > x_i(t_c, u_i')\big) \Rightarrow \big(x_i(t, u_i'') > x_i(t, u_i') \,\forall\, t \geq t_c\big)$.

In the case $x_i(0) \geq a_i$ we consider instead a family

$$\mathcal{U}_d := \cup_{\tau \geq 0} \left\{ u_i \in \mathcal{U}_i : \left\{ \begin{array}{l} u_i := u_{i,max} \,\forall\, t \in [0, \tau] \\ u_i(t) := u_{i,min} \,\forall\, t > \tau. \end{array} \right. \right\}$$

which can be shown to have equivalent properties to the four listed above.

We begin by selecting from the family $\mathcal{U}_{a+b+c}$ or $\mathcal{U}_d$ the greatest input u_i, in the preorder "$\overset{\bar{o}}{\preceq}$", that satisfies $x_i(u_i) \preceq x_j(u_j) - d$ with $J_j \ll J_i$, or simply the greatest input if i has no predecessor. Such an input exists and is unique due to (p.1), (p.2), (p.3), and can be shown to satisfy (5)-(7). If there is no predecessor we set $\bar{u}_i = u_i$, otherwise, if a predecessor j of i exists, $x_i(u_i)$ is tangent to $x_j(u_j) - d$ at some $t \geq 0$ by Proposition 15. In this case we define $\bar{u}_i(t) = u_i(t)$ up to the time of tangency, and $\bar{u}_i(t) = u_j(t)$ afterwards. Then, (i) and (ii) are proven by contradiction using (p.4), which implies that if any u_i' is such that $x(t, u_i') \geq x(t, \bar{u}_i)$ for some $t \geq 0$, such $x(t, u_i')$ must be strictly above $x_j(t, u_j)$ at $t > t'$ and violate (6), while (iii) is a consequence of the construction of $\bar{u}_i$. ■

Proof of Theorem 3: We prove the theorem statement starting from its second part: we consider a vector $\mathbf{u} \in \mathcal{U}$ satisfying (9) and we show that such vector is a maximum of $\bar{\mathcal{U}}(\mathbf{T})$. As a consequence of this, $\bar{\mathcal{U}}(\mathbf{T})$ has maxima.

We begin by noting that $\bar{\mathcal{U}}(\mathbf{T}) \neq \emptyset$ implies $\underline{\mathcal{U}} \neq \emptyset$, since $\bar{\mathcal{U}}(\mathbf{T}) \subset \underline{\mathcal{U}}$. It also implies that $\bar{\mathcal{U}}_i(\emptyset, T_i) \neq \emptyset$ for all i such that $\emptyset \ll J_i$, and that $\bar{\mathcal{U}}_i(x_j, T_i) \neq \emptyset$ for all i such that $J_j \ll J_i$, provided u_j is the j-th component of an input $\mathbf{u} \in \bar{\mathcal{U}}(\mathbf{T})$. This follows by the fact that each component of $\bar{u} \in \mathcal{U}(\mathbf{T})$ satisfies constraints (5), (6), (7). Thus, by Lemma 2 the sets $\bar{\mathcal{U}}_i(\cdot, T_i)$ have maxima and the relation (9) is well defined. Consider a vector $\bar{\mathbf{u}}$ that satisfies (9). Such $\bar{\mathbf{u}} \in \bar{\mathcal{U}}(\mathbf{T})$, since by construction each of its components satisfies (5) and (6). To show that it is maximal we proceed by contradiction. Assume that there exists a $\mathbf{u} \in \bar{\mathcal{U}}(\mathbf{T})$ such that

$$\bar{\mathbf{u}} \overset{\bar{o}}{\not\succeq} \mathbf{u}. \tag{24}$$

We know that for all i such that $\emptyset \ll J_i$ and for all $u_i \in \bar{\mathcal{U}}_i(\emptyset, T_i)$, $\bar{u}_i \overset{\bar{o}}{\succeq} u_i$. Thus, to have (24) there must exist a i and j such that $J_j \ll J_i$, and for some $u_j \in \bar{\mathcal{U}}_j(\cdot, T_j)$ such that $u_j \overset{\bar{o}}{\preceq} \bar{u}_j$, for $\bar{u}_i \in \max_{\overset{\bar{o}}{\preceq}} \bar{\mathcal{U}}_i(x_j(\bar{u}_j), T_i)$, and for $u_i \in \bar{\mathcal{U}}_i(x_j(u_j), T_i)$, $\bar{u}_i \overset{\bar{o}}{\not\succeq} u_i$. This is contradicted by Lemma 2. ■

APPENDIX B
PROOFS OF SECTION IV-C

Sketch of proof of Theorem 7: The theorem is proved by showing that, for all schedules $\mathbf{T}$ that satisfy SP*, $P_i(\mathbf{T}) \leq T_i + \delta_{max}$ for all agents i with $x_i(0) < a_i$. This property ensures that conditions (16)-(20) imply conditions (12) and (13).

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

$$x_i^* := x_i(u_{i,max}, a_i, \dot{x}_{i,min}),$$

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

$$u_i(t) = \left\{ \begin{array}{l} u_{i,max} \text{ for } t \leq \tau_1 \\ u_{i,min} \text{ for } t \in (\tau_1, \tau_2] \\ u_j \text{ for } t > \tau_2 \end{array} \right.$$

with $0 \leq \tau_1 \leq \tau_2 \leq \infty$ and $J_j \ll J_i$, such that

$$x_i(t, u_i) \geq x_i^*(t) - a_i + x_i(0) \,\forall\, t \geq 0 \tag{25}$$

for all $i \in \{1, \ldots, m\}$. Set $u_i(t) = u_{i,max}$ for all t if i has no predecessor. With this input we have $\tau_i(x_i(0)) \geq t_{a_i + d_i^*}(u_i)$ (by (25)) and $x_i(t) \geq \underline{x}_j(t) + d$ for all $t \geq 0$ and $J_j \gg J_i$ (since $\mathbf{u} \in \underline{\mathcal{U}}$).

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 $x_m(T_{m+1}, u_m) \geq x_{m+1}^*(0) + d_{m+1}^*$ and

$$x_m(t + T_{m+1}, u_m) \geq x_{m+1}^*(t) + d_{m+1}^* \,\forall\, t \geq 0. \tag{26}$$

It can be proved that, if $\underline{\mathcal{U}} \neq \emptyset$, (26), and T_i satisfies (16) and (18), then there exists an input u_{m+1} of the form

$$u_{m+1}(t) = \left\{ \begin{array}{l} \underline{u}_i \text{ for } t \leq \tau_1 \\ u_{i,max} \text{ for } t \in (\tau_1, \tau_2] \\ u_{i,min} \text{ for } t \in (\tau_2, \tau_3] \\ u_j \text{ for } t > \tau_3 \end{array} \right.$$

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 $\mathbf{u}$. The resulting $\mathbf{u} \in \bar{\mathcal{U}}(\mathbf{T})$, and by (25) $\tau_i(x_i(0)) \geq t_{a_i + d_i^*}(u_i)$ for all i. Since $\delta_{max} + T_i \geq \tau_i(x_i(0))$ and $P_i(\mathbf{T}) \leq t_{a_i + d_i^*}(u_i)$, we have that $P_i(\mathbf{T}) \leq T_i + \delta_{max}$ for all agents i with $x_i(0) < a_i$, as requested. ■

Proof of Theorem 8: If APPROXIMATESOLUTION returns "*no*" then either $x_i(0) \in [a_i, b_i)$ and $x_j(0) \in [a_j, b_j)$ for some $(i, j) \in \mathcal{I}_+$, or $D_i < R_i$ implying $\underline{\mathcal{U}} = \emptyset$, or POLYNOMIALTIME at line 11 returns "*no*". In the first case, $\mathbf{x}(0) \in B_+ \subset \hat{B}$. In the second case, for all $\mathbf{u} \in \mathcal{U}$, $\mathbf{x}(t, \mathbf{u})$ intersects $B_- \subset \hat{B}$. In the third case, for any $\mathbf{u} \in \underline{\mathcal{U}}$ the schedule $\mathbf{T}$ defined by $T_i = t_{a_i}(u_i)$ satisfies (16) and (20). Since APPROXIMATESOLUTION solves SP* exactly, and finds no feasible schedule, $\mathbf{T}$ must violate either (17), or (18), or (19). If (17) is violated then $x_i(t, u_i) \in [a_i, b_i]$ and $x_j(t, u_j) \in [a_j, b_j]$ for some $t \geq 0$ and for some $(i, j) \in \mathcal{I}_+$ in which case $\mathbf{x}(t, \mathbf{u})$ intersects $B_+ \subset \hat{B}$. If (18) is violated, then $t_{a_i}(u_i) < \tau_j$ for some $J_j < J_i$ with $x_j(0) \geq a_j$ and $x_i(0) < a_i$. We have that $x_j(t_{a_i}(u_i)) \leq x_j(0) + \tau_j \dot{x}_{max}$, and that $\tau_j \leq \delta_{max} - (x_j(0) - a_j)/\dot{x}_{max}$, hence $x_j(t_{a_i}(u_i)) - x_i(t_{a_i}(u_i)) \leq \delta_{max} \dot{x}_{max}$, that is, the trajectory intersects $\hat{B}_-$. If (19) is violated, then $T_i - T_j < \delta_{max}$ for some i, j with $x_i(0) < a_i$, $x_j(0) < a_j$, $T_i \geq T_j$. This implies that $x_i(t_{a_j}(u_j), u_i) -$

$x_j(t_{a_j}(u_j), u_j) \leq \delta_{max}\dot{x}_{max}$, thus, if $(i,j) \in \mathcal{I}_+$ the trajectory intersects $\hat{B}_+$, if $(i,j) \in \mathcal{I}_-$ it intersects $\hat{B}_-$. ■

Proof of Theorem 9: We prove the theorem by showing that, if $\gamma_+ < 1$ or $\gamma_- < 1$, we can select $\{\mathbf{x}(0), \dot{\mathbf{x}}(0), \Theta\}$ such that APPROXIMATESOLUTION returns "no" and construct a trajectory that does not intersect $\check{B}$. Consider first $\gamma_+ < 1$. Take a system with two agents on two different paths. Assume $\dot{x}_{1,max} = \dot{x}_{2,max} := \dot{x}_{max}$, $\dot{x}_{1,min} = \dot{x}_{2,min} := \dot{x}_{min}$, $b_1 - a_1 = b_2 - a_2 := b - a$. Let $x_1(0) < a_1$, $x_2(0) < a_2$, $x_1(0) > x_2(0)$, $\dot{x}_1(0) = \dot{x}_{max}$, and $\dot{x}_2(0) = \dot{x}_{min}$. 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 $\epsilon > 0$. APPROXIMATESOLUTION returns "no", since $t_{a_1}(u_{1,max}) = R_1$, $t_{a_2}(u_{2,min}) = D_2$, $R_1 < R_2$, and $D_1 < D_2$. Consider the trajectory $\mathbf{x}(t) := (x_1(t, u_{i,max}), x_2(t, u_{2,min}))$. We have $x_1(t_{a_2}(u_{2,min}), u_{1,max}) = a + (\delta_{max} - \epsilon)\dot{x}_{max}$, thus $\mathbf{x}(t) \notin \check{B}$ for all $t \geq 0$ provided that $\epsilon \in \big(0, \delta_{max}(1 - \gamma_+)\big)$.

Now consider $\gamma_- < 1$ and consider the same system as above, with identical parameters but with agents 1 and 2 on the same path. APPROXIMATESOLUTION returns "no' for the same reasons as above. Consider the trajectory $\mathbf{x}(t) := (x_1(t, u_{1,max}), x_2(t, u_{2,min}))$. We have that $|x_1(t, u_{1,max}) - x_2(t, u_{2,min})| \geq |x_1(0) - x_2(0)| = (\delta_{max} - \epsilon)\dot{x}_{max} - (\dot{x}_{max} - \dot{x}_{min})t_{a_2}(u_{2,min})$. Thus, $\mathbf{x}(t) \notin \check{B}$ for all $t \geq 0$ provided that $\epsilon \in \Big(0, \delta_{max}(1 - \gamma_-) - t_{a_2}(u_{2,min})\big(1 - \frac{\dot{x}_{min}}{\dot{x}_{max}}\big)\Big)$, which is nonempty for $t_{a_2}(u_{2,min}) < \delta_{max}(1 - \gamma_-)\dot{x}_{max}/(\dot{x}_{max} - \dot{x}_{min})$ ■

ACKNOWLEDGMENT

This work was supported by the NSF Award # CNS 0930081

REFERENCES

[1] R. M. V. Auken, J. Zellner, D. P. Chiang, J. Kelly, J. Y. Silberling, R. Dai, P. C. Broen, A. M. Kirsch, and Y. Sugimoto, "Advanced crash avoidance technologies (ACAT) program – final report of the Honda-DRI team, volume I: Executive summary and technical report." U.S. Department of Transportation, National Highway Traffic Safety Administration, Tech. Rep., 2011.

[2] Z. R. Doerzaph, V. L. Neale, J. R. Bowman, D. C. Viita, and M. Maile, "Cooperative intersection collision avoidance system limited to stop sign and traffic signal violations (CICAS-V) Subtask 3.2 interim report: Naturalistic infrastructurebased driving data collection and intersection collision avoidance algorithm development," National Highway Traffic Safety Administration, Tech. Rep., 2008.

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

[4] F. Basma, Y. Tachwali, and H. Refai, "Intersection collision avoidance system using infrastructure communication," in *14th International IEEE Conference on Intelligent Transportation Systems*, 2011.

[5] M. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst*, 2013.

[6] Kyoung-Dae Kim, "Collision free autonomous ground traffic: A model predictive control approach," in *International Conference on Cyber-Physical Systems*, 2013.

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

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

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

[10] C. Tomlin, J. Lygeros, and S. Sastry, "Synthesizing controllers for nonlinear hybrid systems," in *Hybrid systems: Computation and control*, 1998.

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

[12] R. Ghosh and C. Tomlin, "Maneuver design for multiple aircraft conflict resolution," in *American Control Conference*, 2000.

[13] C. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proc. IEEE*, vol. 88, pp. 949–970, 2000.

[14] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Trans. Intell. Transp. Syst*, vol. 2, pp. 110–120, 2001.

[15] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proc. IEEE*, vol. 91, pp. 986–1001, 2003.

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

[17] A. Colombo and D. Del Vecchio, "Enforcing safety of cyberphysical systems using flatness and abstraction," in *Proceedings of the Work-in-Progress session of ICCPS*, 2011.

[18] M. R. Hafner, D. Cunningham, L. Caminiti, and D. D. Vecchio, "Automated vehicle-to-vehicle collision avoidance at intersections," in *Proc. of ITS World Congress*, 2011.

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

[20] E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune, "Supervisory control for collision avoidance in vehicular networks using discrete event abstractions," in *American Control Conference*, 2013.

[21] ——, "Supervisory control for collision avoidance in vehicular networks with imperfect measurements," in *IEEE Conference on Decision and Control*, 2013.

[22] I. M. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by hamilton-jacobi projections," *Journal of Scientific Computing*, vol. 19, pp. 323–346, 2003.

[23] A. Girard, C. L. Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Hybrid Systems: Computation and Control*, 2006.

[24] M. Althoff, C. Le Guernic, and B. H. Krogh, "Reachable set computation for uncertain time-varying linear system," in *Hybrid systems: Computation and Control*, 2011.

[25] O. Shakernia, G. J. Pappas, and S. Sastry, "Decidable controller synthesis for classes of linear systems," in *Hybrid Systems: Computation and Control*, 2000.

[26] O. Shakernia, G. J. Pappas, and S. S. Sastry, "Semi-decidable synthesis for triangular hybrid systems." in *Hybrid Systems: Computation and Control*, 2001.

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

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

[29] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution for air traffic management systems," *IEEE Trans. Intell. Transp. Syst*, vol. 1, pp. 221–232, 2000.

[30] L. Pallottino, E. M. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Trans. Intell. Transp. Syst*, vol. 3, pp. 3–11, 2002.

[31] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *J. Guid. Control Dynam.*, vol. 25, pp. 755–764, 2002.

[32] Z.-H. Mao, D. Dugail, and E. Feron, "Stability of intersecting aircraft flows using heading-change maneuvers for conflict avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, pp. 357–369, 2005.

[33] M. A. Christodoulou and S. G. Kodaxakis, "Automatic commercial aircraft-collision avoidance in free flight: the three-dimensional problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, pp. 242–249, 2006.

[34] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. Robot.*, vol. 23, pp. 1170–1183, 2007.

[35] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Trans. Veh. Technol.*, vol. 60, pp. 804–818, 2011.

[36] G. R. de Campos, P. Falcone, and J. Sjöberg, "Autonomous cooperative driving: a velocity-based negotiation approach for intersections crossing," in *IEEE Conference on Intelligent Transportation Systems*, 2013.
[37] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Trans. Intell. Transp. Syst*, vol. 1, pp. 199–220, 2000.
[38] M. Prandini and J. Hu, "Application of reachability analysis for stochastic hybrid systems to aircraft conflict prediction," *IEEE Trans. Autom. Control*, vol. 54, pp. 913–917, 2009.
[39] M. Prandini, V. Putta, and J. Hu, "Air traffic complexity in future air traffic management systems," *Journal of Aerospace Operations*, vol. 1, pp. 281–299, 2012.
[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.
[41] J. K. Lenstra, A. H. G. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of discrete mathematics*, vol. 1, pp. 343–362, 1977.
[42] M. R. Garey, D. S. Johnson, B. B. Simons, and R. E. Tarjan, "Scheduling unit-time tasks with arbitrary release times and deadlines," *SIAM J. Comput.*, vol. 6, pp. 416–426, 1981.
[43] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge University Press, 2002.
[44] R. Verma, D. D. Vecchio, and H. K. Fathy, "Development of a scaled vehicle with longitudinal dynamics of an HMMWV for an ITS testbed," *IEEE/ASME Transactions on Mechatronics*, vol. 13, pp. 1–12, 2008.
[45] Attached Supplementary Material, will be available on authors website upon publication.
[46] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *IEEE Conference on Decision and Control*, 2013.