A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections

Alessandro Colombo
DEIB, Politecnico di Milano,
via Ponzio 34/5, 20133 Milano, Italy.
alessandro.colombo@polimi.it

***Abstract*—Active safety systems for collision avoidance of road vehicles are a promising solution to the enormous human and monetary cost of road crashes. In this paper, we describe some new mathematical approaches for the construction of collision avoidance algorithms for human-driven vehicles. These algorithms exploit the sensing and communication capabilities of the vehicles to share information on the current state of the system, and based on this information check the safety of the human decisions and correct them when they would lead to a collision.**

I. Introduction

According to the latest European Road Safety Observatory annual statistical report [1], road traffic accidents in the European Union cause about 37000 deaths and more than 1.2 million injuries per year, with an estimated annual cost of €145 billion. Similar figures are reported abroad. Interestingly, about 25% of all road fatalities reported in [1] have taken place at road junctions, suggesting that crashes at traffic intersections constitute a very significant portion of the total number of crashes and fatalities. The ultimate causes of such crashes are often hard to ascertain, but it is generally estimated that between 60% and 95% of all crashes are due to human error. Clearly, even a marginal reduction of the chances of human mistake can have a dramatic impact on the human and economic costs of traffic collisions.

Even though much has been done in recent years to reduce the causes and effects of human error, mostly in terms of road regulation and vehicle passive safety, the recent appearance of cheap and powerful computation, communication, and sensing devices is paving the way towards a more effective solution, based on technology. The most visible attempts in this direction are probably the vehicles of the DARPA challenges and the Google car which, by completely removing the driver from the equation, cut the problem at its roots. However, the likeliness of having fully autonomous cars replace the current human-driven ones any time soon is debatable [2], if not for the technological challenge for the marketing and regulatory problems that would ensue. A more likely solution will thus see, at least in the short run, the use of technology as a support to the human driver, rather than as a replacement [3]–[7]. Unfortunately, as we see next, the presence of a human driver introduces an extra degree of mathematical complexity in the collision avoidance problem.

The mathematical structure of an algorithm to check the correctness of human behaviour (or of any process) according to given rules was well formalised in [8] in the framework of discrete event processes. Such an algorithm is called a *supervisor*. In our case, the supervisor's task is to check that the drivers do not drive the system (the aggregate set of all cars and other agents involved in the supervisory problem) into a state of collision, or in a state that cannot but result in a future collision. Such a problem, involving state reachability, is called a *supervisory problem with safety specification*. Note that, while computing the set of collision states (which is typically called the *bad set*) is a relatively simple task ultimately reducible to checking a set of inequalities, the computation of the states that are bound to lead to a collision (which is called the *capture set*, in analogy to a similar problem in differential game theory) is a much harder task requiring to evaluate all the possible trajectories that a given state might follow under the set of allowed input signals.

An important property of a supervisor that must be implemented in real vehicles is to be *minimally restrictive*, that is, to prevent only those choices of manoeuvre that will actually cause a collision, avoiding false alarms. As we explain next, the synthesis of an exact minimally restrictive supervisor is a computationally intractable problem when many vehicles are involved. With some care it can be solved for small but relevant scenarios, while an approximate solution can be computed in realistic conditions.

An other paramount ingredient of our problem is cooperativity, that is, the capacity of vehicles to share information and to coordinate. Indeed, with suitable initial conditions, the adversarial game of collision avoidance of two non-cooperating agents always has a winning strategy for the collision-seeking one. A simple example is found letting the two agents have identical dynamics and initial conditions and moving on intersecting paths: one agent can cause a collision simply by mirroring the moves of the other. Even though such a scenario is typically unreasonable, it suggest how an exact supervisor for a non-cooperative scenario can rapidly become very restrictive, and some degree of communication and cooperation is to be expected from most if not all involved agents in order to have a provably safe, yet not overly restrictive architecture.

This paper outlines some results presented in [9], [10], and extends them to a network with arbitrarily many intersections. In the following sections, we discuss a mathematical framework to construct supervisors with the properties identified above. In Section II we formalise a *verification problem* which is at the heart of the design of a supervisor, in Section III we show how the verification problem can be translated into a

978-1-4799-5863-4/14/$31.00 © 2014 IEEE

scheduling equivalent in the case of a single intersection, and in Section IV we sketch how this result can be extended to an arbitrarily large road network.

II. Problem formulation

Our objective is to avoid collisions of vehicles moving along intersecting roads. The primary means of control are the longitudinal actuators (brakes and engine torque), while the vehicle's lateral dynamics essentially acts as a disturbance on the longitudinal model (and is not considered in the following equations). Accordingly, we model agents as Newtonian point masses moving along pre-specified paths, with possible intersections between these paths. Each agent's dynamics is described by the equation

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

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. For simplicity we assume that all agents have the same dynamics, though this constraint can be relaxed for agents on different paths (see [9], [10]). We will use the symbols x_i and u_i both for signals (functions of time) and signal values (vectors in a Euclidean space), the difference should be clear from the context. When the dependence of x_i on the input, time, or initial conditions needs to be made explicit, we list the independent quantities as an argument (e.g. $x_i(u_i)$ is the trajectory x_i with input u_i, $x_i(t, u_i, x_i(t_0), \dot{x}_i(t_0))$ is the state x_i at time t reached with input u_i from initial state $(x_i(t_0), \dot{x}_i(t_0))$). Boldface $\mathbf{x}$ and $\mathbf{u}$ denote the vectors $(x_1, x_2, ...)$, $(u_1, u_2, ...)$.

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. We make the following assumptions:

(A.1) U_i has a unique minimum u_{min} and a unique maximum u_{max}, and $f(\dot{x}_i, u_i)$ 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 positive interval $[0, \dot{x}_{max}]$ with nonempty interior.

(A.4) $|f(\dot{x}_i, u_i)|$ is bounded for all $\dot{x}_i \in [0, \dot{x}_{i,max}]$, $u_i \in U_i$,

(A.5)
$$\begin{array}{c} \lim_{t\to\infty} \dot{x}_i(t, u_{max}) = \dot{x}_{max} \\ \lim_{t\to\infty} \dot{x}_i(t, u_{min}) = 0 \end{array} \tag{2}$$

(min and max velocities are attained at least asymptotically by applying u_{min} and u_{max}),

As shown in [11], (A.1) implies the following monotonicity property

$$\begin{array}{c} (x_i(0), \dot{x}_i(0)) \geq (x'_i(0), \dot{x}'_i(0)), u_i(t) \geq u'_i(t)\, \forall\, t \geq 0 \\ \Downarrow \\ (x_i(t), \dot{x}_i(t)) \geq (x'_i(t), \dot{x}'_i(t))\, \forall\, t \geq 0. \end{array}$$

Let $\mathcal{I}_k$ be the k-th intersection in the network, and $\mathcal{P}_h$ be the h-th path (see e.g., Fig. 1). We assume all paths to be closed and non-circular, that is, they have a beginning and an end. Let $\mathcal{P}_h \in \mathcal{I}_k$ mean that path $\mathcal{P}_h$ crosses intersection $\mathcal{I}_k$, and $i \in \mathcal{P}_h$ mean that agent i lays on path $\mathcal{P}_h$. We say that an intersection $\mathcal{I}_{k_1}$ is *downstream* an intersection $\mathcal{I}_{k_2}$ along a path $\mathcal{P}_h$ if $\mathcal{I}_{k_2}$ follows $\mathcal{I}_{k_1}$ along the traffic flow of path $\mathcal{P}_h$. Given a set of labelled agents along one or multiple paths, we say that they are labelled in *topological order* if, whenever agents i and j lay on the same path and i drives behind j, then $i > j$. In the following we assume that each agent's path is known ahead, and that paths can intersect but never merge. This is of course a rather restrictive assumption, and its relaxation is currently under study.

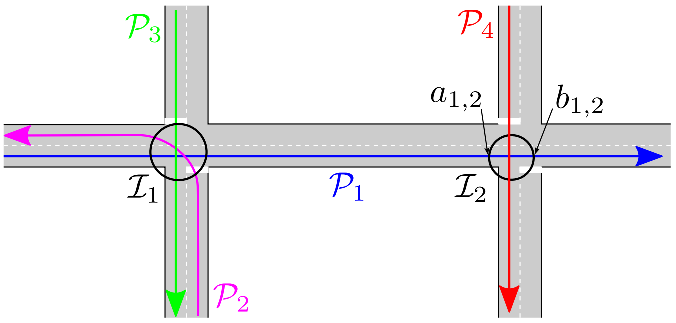

Fig. 1. Two intersections ($\mathcal{I}_1$ and $\mathcal{I}_2$) of four paths ($\mathcal{P}_1$ to $\mathcal{P}_4$). In the figure, $\mathcal{I}_2$ is downstream $\mathcal{I}_1$ along $\mathcal{P}_1$. The interval $[a_{1,2}, b_{1,2}]$, representing the length of the intersection $\mathcal{I}_2$ along path $\mathcal{P}_1$, is represented as an example.

For each $\mathcal{P}_h \in \mathcal{I}_k$ we consider an interval $[a_{h,k}, b_{h,k}]$, which defines the length of the intersection $\mathcal{I}_k$ along the path $\mathcal{P}_h$ (see Fig. 1). A side impact occurs if two agents from different paths enter the respective intervals $[a, b]$ simultaneously, while a rear-end collision occurs if two agents on the same path have distance less than d, which we assume to be fixed for all agents. The bad set B is the set of all collision states, given by the union of the sets

$$\begin{array}{l} B_+ := \{\mathbf{x} \in \mathbb{R}^n : \exists (i, j, h_1, h_2, k); i \in \mathcal{P}_{h_1}, j \in \mathcal{P}_{h_2}, \\ \mathcal{P}_{h_1}, \mathcal{P}_{h_2} \in \mathcal{I}_k; x_i \in (a_{h_1,k}, b_{h_1,k});\ \text{and}\ x_j \in (a_{h_2,k}, b_{h_2,k})\} \end{array}$$

which accounts for all side impacts, and

$$B_- := \{\mathbf{x} \in \mathbb{R}^n : \exists (i, j, k); i \in \mathcal{P}_h, j \in \mathcal{P}_h; |x_i - x_j| < d\},$$

which accounts for all rear-end collisions.

Given the above assumptions, the key step of the supervisory algorithm, that of verifying whether a given configuration of agents is bound to evolve into a collision, is formally stated in the following verification problem.

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

The above is a *decision problem*, with instances described by the set $(\mathbf{x}, \dot{\mathbf{x}})$ of initial conditions (and by all parameters of the system, though for simplicity we assume these fixed once and for all). Using a common notation from complexity theory, we write $(\mathbf{x}, \dot{\mathbf{x}}) \in \text{VP}$ to mean that Problem VP *accepts* the instance $(\mathbf{x}, \dot{\mathbf{x}})$, that is, it finds a safe input signal. Along the same line, we use the notation $P1 \simeq P2$ to mark that two decision problems $P1$ and $P2$ are *equivalent*, meaning that they share the same set of instances, and accept the same subset of instances (a more formal and general definition of equivalence is given e.g. in [12], but the one given here suffices for our purposes).

Problems similar to the one above have been long studied in the hybrid systems literature, and elegant general purpose solutions have been proposed using a Hamilton-Jacobi formulation of VP, or a discretization and representation of the system as a discrete event abstraction [13]–[22]. These approaches unfortunately are not suitable to handle large road networks due to the curse of dimensionality. This is not surprising, and VP was formally shown to be NP-hard in some significant cases in [9], [23]. However, in the next sections we present an equivalent formulation of VP which allows to express the problem in a compact fashion, focussing on the combinatorial structure and discarding the dynamic complexity. This allows to solve in real time (i.e. fractions of a second) problems with up to seven agents, and to devise approximate algorithms suitable to solve much larger instances.

III. Scheduling formulation of the verification problem

To begin with, assume that we have a single intersection, $\mathcal{I}_k$, n intersecting paths, and a single agent on each path. We introduce the following scheduling quantities. Assume that a set $(\mathbf{x}, \dot{\mathbf{x}})$ of initial conditions is given. For sake of clarity, we do not include them among the arguments of the following functions. Let $t_\alpha(u_i) := \min\{t \geq 0 : x_i(t, u_i) \geq \alpha\}$ and $t_\alpha(u_i) := \infty$ if $x_i(t, u_i) < \alpha$ for all $t \geq 0$, that is, $t_\alpha(u_i)$ is the earliest time when agent i can pass the point α. Given $i \in \mathcal{P}_h \in \mathcal{I}_k$, let

$$\begin{array}{rcl} R_{k,i} & := & t_{a_{h,k}}(u_i) \text{ with } u_i = u_{max}, \\ D_{k,i} & := & t_{a_{h,k}}(u_i) \text{ with } u_i = u_{min}. \end{array} \tag{3}$$

These are the earliest and latest time when agent i can reach $a_{h,k}$. Now, given a vector $\mathbf{T}_k = (T_{k,1}, \dots, T_{k,n})$, let $\bar{\mathcal{U}}(\mathbf{T}_k)$ be the subset of $\mathcal{U}$ of inputs such that

$$i \in \mathcal{P}_h \in \mathcal{I}_k, x_i(0) \leq a_{h,k} \Rightarrow x_i(t, u_i) \leq a_{h,k} \, \forall \, t < T_{k,i} \tag{4}$$

for all i. In other words, the set $\bar{\mathcal{U}}(\mathbf{T}_k)$ contains all the input signals such that the trajectory x_i does not cross $a_{h,k}$ before $T_{k,i}$. Then, define the quantity $P_i(\mathbf{T}_k)$ as follows. If $\bar{\mathcal{U}}(\mathbf{T}_k) = \emptyset$, $P_i(\mathbf{T}_k) := \infty$ for all $i \in \{1, \dots, n\}$, otherwise take

$$P_i(\mathbf{T}_k) := \inf_{u_i \in \bar{\mathcal{U}}(\mathbf{T}_k)} t_{b_i}(u_i). \tag{5}$$

$P_i(\mathbf{T}_k)$ is the earliest time when i can reach $b_{h,k}$, avoiding rear end collisions, if it does not pass $a_{h,k}$ before $T_{k,i}$.

We can now introduce the scheduling problem.

SP$_k$ *Given initial conditions $(\mathbf{x}, \dot{\mathbf{x}})$, determine if there exists a schedule $\mathbf{T}_k \in \mathbb{R}^n_+$ such that for all $i \in \{1, \dots, n\}$*

$$R_{k,i} \leq T_{k,i} \leq D_{k,i}, \tag{6}$$

and for all $i \in \mathcal{P}_{h_1}, j \in \mathcal{P}_{h_2}; \mathcal{P}_{h_1}, \mathcal{P}_{h_2} \in \mathcal{I}_k$; if $x_i(0) < b_i$, then

$$T_{k,i} \geq T_{k,j} \Rightarrow T_{k,i} \geq P_j(\mathbf{T}_k). \tag{7}$$

Theorem 1 *VP $\simeq$ SP$_k$.*

Proof: The theorem was proved in [9] for the case of $\dot{x}$ is strictly positive. The extension for $\dot{x} \in [0, \dot{x}_{max}]$ is trivial. ■

The above theorem states that SP$_k$ is equivalent to VP, meaning that by solving SP$_k$ we solve VP exactly, and *vice versa*. But whereas VP is formulated over the functional space $\mathcal{U}$, SP$_k$ is defined over the Euclidean space $\mathbb{R}^n$. The search space can be further reduced using the following result.

Lemma 2 *If SP$_k$ has a feasible schedule $\mathbf{T}_k$, then it has a feasible schedule $\mathbf{T}'_k$ where*

$$T'_{k,i} \geq T'_{k,j} \Rightarrow T'_{k,i} = P_j(\mathbf{T}_k). \tag{8}$$

Proof: Condition (7) requires that intervals $[T_{k,i}, P_i(\mathbf{T}_k)]$, $[T_{k,j}, P_j(\mathbf{T}_k)]$ do not intersect, and $P_i(\mathbf{T}_k)$ is nonincreasing for decreasing $T_{k,i}$. Thus, if there is a feasible schedule $\mathbf{T}_k$, associated to a sequence of nonintersecting segments, a new feasible schedule can be obtained by shifting the left-end of all segments to satisfy (8). ■

Through this lemma we obtain a finite state space, corresponding to the set of all possible orderings of the n agents. For few agents, the verification problem can be solved by a simple exhaustive search, while when many agents are involved approximations with a guaranteed error bound can be constructed using results from the scheduling literature (see e.g., [9], [10], [24], [25]). The above results, and in particular the equivalence in Theorem 1, can be extended to the case when more than one agent lies on each path, by requiring the condition

$$\begin{array}{c} i, j \in \mathcal{P}_h \text{ and } x_i(0) < x_j(0) \Rightarrow \\ x_i(t, u_i) \leq x_j(t, u_j) - d \, \forall \, t \geq 0 \end{array}$$

to hold along with the condition (4) for all inputs in $\bar{\mathcal{U}}(\mathbf{T}_k)$. A full proof of this will appear in a forthcoming paper.

Once the solution of VP is computed, a supervisor can be implemented simply by measuring the current state of the system (e.g., using onboard positioning and speed sensors and sharing the information among all nearby vehicles), projecting the state of the system forward according to the input currently requested by all drivers, and checking that the reached state lies outside of the capture set. This process is formalised in Algorithm 1, where $\mathbf{v}_k$ is the vector of inputs requested by all drivers at time $k\tau$. The input $\mathbf{u}_{safe}$ returned by Algorithm 1

Algorithm 1 Implementation of the supervisor map

```
1: procedure $s(\mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau), \mathbf{v}_k)$
2:     $\bar{\mathbf{u}}(t) \leftarrow \mathbf{v}_k \, \forall \, t \in [k\tau, (k+1)\tau]$
3:     $\mathbf{x}_{k+1} \leftarrow \mathbf{x}((k+1)\tau, \bar{\mathbf{u}}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
4:     $\dot{\mathbf{x}}_{k+1} \leftarrow \dot{\mathbf{x}}((k+1)\tau, \bar{\mathbf{u}}, \mathbf{x}(k\tau), \dot{\mathbf{x}}(k\tau))$
5:     $(\mathbf{x}_{k+1}, \dot{\mathbf{x}}_{k+1}) \in$ VP?
6:     if $(answer = yes)$ and $\mathbf{x}(t, \bar{\mathbf{u}}) \notin B$ for all $t \in [k\tau, (k+1)\tau]$ then
7:         return $\bar{\mathbf{u}}$
8:     else
9:         return $\mathbf{u}_{safe}$
```

when the desired input is deemed unsafe can be computed as the solution of a linear-complexity optimal control problem, forcing the agents to respect any schedule that is feasible at time $k\tau$. One such schedule is guaranteed to exist, and is known as a by-product of the past iteration of Algorithm 1. The trajectories in Fig. 2 were obtained simulating a set of 7 vehicles at the intersection of three paths. We used the dynamic model

$$\ddot{x}_i = \begin{cases} u_i - 0.0005\dot{x}_i^2 \text{ if } (\dot{x}_i > 0, u_i - 0.0005\dot{x}_i^2 \leq 0) \\ \quad \text{or } (\dot{x}_i < 0, u_i - 0.0005\dot{x}_i^2 \geq 0) \\ 0 \text{ otherwise,} \end{cases}$$

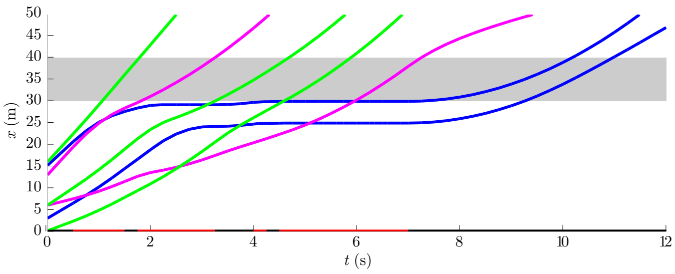

Fig. 2. Simulation with 7 agents on 3 paths intersecting at a single point (as in intersection $\mathcal{I}_1$ of Fig. 1). Trajectories of the same colour belong to agents on the same path. The gray rectangle represents the intersection interval along the paths; coordinates were chosen to have the three intervals coincide at $x \in [30, 40]$. A side-impact occurs if two trajectories with different colours lie in the gray rectangle simultaneously, a rear-end collision if two trajectories of the same colour are too close. The t-axis is black when the supervisor is applying the drivers' desired input, red when it is overriding it. In its current form, for the sake of simplicity, the supervisor overrides the inputs of all vehicles simultaneously.

where the quadratic term accounts for air drag, with $\dot{x} \in [0, 50\,km/h]$ and $u \in [-8\,m/s^2, 2\,m/s^2]$. The drivers were assigned a given fixed speed that they try to maintain, and the supervisor acted to correct the driver's input to avoid side impacts (two trajectories of different colours simultaneously in the gray rectangle in Fig. 2) and rear-end collisions (two trajectories of the same colour at a distance less than $d = 5m$.

IV. Extensions to many intersections

The results in the previous section prove that, in a network with a single intersection, VP can be rewritten as the scheduling problem SP$_k$, which has a finite search space. Its solution can then be found by an enumerative algorithm or through other heuristics or approximations, and can be used to construct a least restrictive supervisor for cooperative collision avoidance. In the following we prove that this approach can be extended to a generic network comprising an arbitrary number of intersections, provided a *sparsity condition* holds. This condition ensures that VP for vehicles near different intersections can be decoupled, and each intersection can be handled as if it was isolated. To define such a condition, we begin by computing the minimum worst-case stopping distance of an agent:

Definition 1 $D_{stop} := \lim_{t\to\infty} x_i(t, u_{min})$ *with* $x_i(0) = 0$, $\dot{x}_i(0) = \dot{x}_{max}$.

We have the following.

Definition 2 (Sparse traffic condition) *Traffic is sparse if, given any two intersections* $\mathcal{I}_{k_1}$, $\mathcal{I}_{k_2}$ *along a path* $\mathcal{P}_h$, *with* $\mathcal{I}_{k_2}$ *downstream* $\mathcal{I}_{k_1}$, *and given the distance* $L := a_{h,k_2} - b_{h,k_1} \geq D_{stop}$ *between the two intersections, there are no more than* $(L - D_{stop})/d + 1$ *agents lying between* $a_{h,k_1} - D_{stop}$ *and* a_{h,k_2}.

Now, we design a map that, given a road network and a set of agents, assigns each agent to one and only one intersection.

Definition 3 (traffic assignment) *A traffic assignment is a map* $\{1, \ldots, n\} \overset{ta}{\to} \{\mathcal{I}_{k_1}, \mathcal{I}_{k_m}\}$ *which maps agents to intersections as follows:*

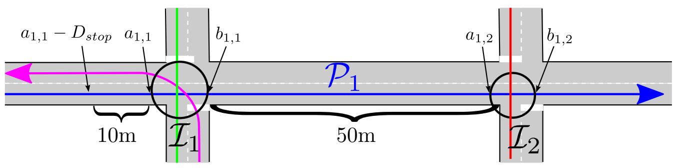

Fig. 3. In the network of Fig. 1, assuming $D_{stop} = 10$m and $d = 5$m, traffic is sparse provided there are no more than 9 vehicles between $a_{1,1} - D_{stop}$ and $a_{1,2}$ on path $\mathcal{P}_1$

Algorithm 2 traffic assignment

```
1: for all segments of a path $\mathcal{P}_h$ between two subsequent intersections $\mathcal{I}_{k_1}$ and $\mathcal{I}_{k_2}$, with $\mathcal{I}_{k_2}$ downstream $\mathcal{I}_{k_1}$ do
2:     $m \leftarrow$ Number of agents between $a_{h,k_1} - D_{stop}$ and $b_{h,k_1}$
3:     assign labels $\{1, \ldots n\}$ to agents between $b_{h,k_1}$ and $a_{h,k_2}$, in reverse topological order
4:     for $i = 1 \to n$ do
5:         if $\lim_{t\to\infty} x_i(t, u_{min}) \geq D_{stop} + (m + i - 1)d$ then
6:             assign $\{1, \ldots, i-1\}$ to $\mathcal{I}_{k_1}$ and $\{i \ldots, n\}$ to $\mathcal{I}_{k_2}$
7:             return
8:     If we reach this line, assign $\{1, \ldots, n\}$ to $\mathcal{I}_{k_1}$
9:     return
```

Definition 4 *Given a traffic assignment* $\overset{ta}{\to}$, *call* $\overset{\overset{ta}{\to}k}{(\mathbf{x}, \dot{\mathbf{x}})}$ *the subset of the elements of vector* $(\mathbf{x}, \dot{\mathbf{x}})$ *relative to agents assigned to intersection* $\mathcal{I}_k$.

With sparse traffic, the traffic assignment in Algorithm 2 partitions agents between intersections so that the verification problem VP can be solved simply by solving, for all k, the scheduling problem SP$_k$ for the agents assigned to $\mathcal{I}_k$ as if it was isolated. This is formally stated in the following theorem.

Theorem 3 *Consider a network with sparse traffic. Then* $(\mathbf{x}, \dot{\mathbf{x}}) \in$ *VP if and only if* $\overset{\overset{ta}{\to}k}{(\mathbf{x}, \dot{\mathbf{x}})} \in$ *SP*$_k$ *for all* k .

Proof: (Sketch) The implication $(\mathbf{x}, \dot{\mathbf{x}}) \in$VP$\Rightarrow \overset{\overset{ta}{\to}k}{(\mathbf{x}, \dot{\mathbf{x}})} \in$ SP$_k$ for all k follows from Theorem 1, considering one intersection at a time. Now consider any two intersections $\mathcal{I}_{k_1}$ and $\mathcal{I}_{k_2}$, with the latter downstream the former along a path $\mathcal{P}_h$. Let i be the last agent in topological order assigned to $\mathcal{I}_{k_2}$, and j be the first agent in topological order assigned to $\mathcal{I}_{k_1}$, along the path $\mathcal{P}_h$. To prove the converse implication one can show that the traffic assignment partitions agents so that, if $\overset{\overset{ta}{\to}k}{(\mathbf{x}, \dot{\mathbf{x}})} \in$ SP$_k$, there exists an input $\mathbf{u}$ such that agents assigned to $\mathcal{I}_{k_1}$ do not collide among themselves, j stops before reaching a_{h,k_2}, and for all $\mathbf{u}' \in \mathcal{U}$, $x_i(t, u'_i) \geq x_j(t, u_j)$ for all $t \geq 0$, that is, agents j and i do not collide regardless of the input chosen by i. Since this holds for all intersections downstream $\mathcal{I}_k$, we can construct an input $\mathbf{u} \in \bar{\mathcal{U}}$ by composing inputs $\mathbf{u}' \in \bar{\mathcal{U}}_k$ for all k, therefore $\overset{\overset{ta}{\to}k}{(\mathbf{x}, \dot{\mathbf{x}})} \in$ SP$_k$ for all k implies $(\mathbf{x}, \dot{\mathbf{x}}) \in$ VP. ■

V. Conclusion

We have presented an approach to cooperative collision avoidance for human-driven vehicles. In the framework discussed above, all vehicles in a road network are assumed to communicate and share an exact knowledge of their state. The results presented here improve over the current state of the art, in particular extending the results in [9], [10], allowing to solve the verification problem (VP) exactly over a network with an arbitrary number of intersections, provided that these verify a sparsity condition. Moreover, some of the assumptions taken here have been at least partially relaxed, albeit in a simpler scenario with a single intersection and no rear-end collisions, in [24] (where the supervisor deals with noisy measurements and control inputs) and [25] (where uncontrollable vehicles are considered).

There are, however, a number of important limitations that still need to be addressed if this approach is to be applied to a real intelligent transport system. A first and major one is the geometry of the paths and intersections: even assuming that the intended path of each car can be determined or reasonably estimated, the current approach does not consider merging or splitting paths, nor it can handle the clusters of nearby intersections that the full set of possible paths on a road network such as that in Fig. 1 would generate (sparsity poses a lower bound on the distance between intersections). Work to extend the above framework to such topologies is under way. A second important limitation is the fact that, currently, the supervisor is designed as a centralised entity and acts simultaneously on all agents. While this might be reasonable for a small set of vehicles near an intersection (here the supervisor could be an algorithm running on road-side infrastructure, or shared among the vehicles), it certainly isn't so for a large road network. The computation has to be distributed and based on local information in order to be applicable in reality, and the communication constraints and delays, the limited sensing capabilities, and the unknown and unpredictable human decision regarding, e.g., the desired path will all have to be taken into account. This is certainly a challenging task, but the results we have obtained so far seem to suggest that it can be completed.

References

[1] (2012) European road safety observatory (ERSO) annual statistical report. http://ec.europa.eu/transport/road_safety/pdf/statistics/dacota/dacota-3.5-asr-2012.pdf.

[2] http://www.technologyreview.com/review/513531/proceed-with-caution-toward-the-self-driving-car/.

[3] 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.

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

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

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

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

[8] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Opt.*, vol. 25, pp. 206–230, 1987.

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

[10] ——, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, (to appear).

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

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[13] 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.

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

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

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

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

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