Robust multi-agent collision avoidance through scheduling

Robust multi-agent collision avoidance through scheduling

Leonardo Bruni, Alessandro Colombo, Domitilla Del Vecchio

***Abstract*— We propose a class of computationally efficient algorithms for conflict resolution in the presence of modeling and measurement uncertainties. Specifically, we address a scenario where a number of agents, whose dynamics are possibly nonlinear, must cross an intersection avoiding collisions. First, we solve the problem of checking membership in the set of states for which there exists an input signal that avoids collisions for any possible disturbance and measurement error. Then, we use this solution to design a supervisor for collision avoidance which is robust to disturbances and measurement uncertainties. We obtain an exact solution and an approximate one with quantified error bound and whose complexity scales polynomially with the number of agents.**

I. INTRODUCTION

The wide diffusion of partly or fully automated agents in disparate engineering applications, from transport systems to production lines, has made the coordinated control of such agents a very interesting topic for control theory. The problem of avoiding conflict configurations when human operators are present in the control loop has proved especially challenging, since this case presents the added constraint of ensuring a conflict-free behaviour without restricting unnecessarily the human commands, casting the problem in the form of a least restrictive supervisory control problem [1]. In the hybrid systems literature, this problem is typically solved by computing the *Maximal Controlled Invariant Set* [2], [3], [4], which is the largest set of states that admit an input that avoids conflicts for all positive times. In this paper, we design a least restrictive supervisor for collision avoidance between a number of agents following intersecting paths. Specifically, we consider a set of agents (such as cars, trains, or airplanes) moving along predetermined paths that intersect at a unique point. The conflict resolution problem for this scenario has been solved in [5] for the case of perfect information, and in the absence of any disturbance. However, this is an unrealistic assumption in almost all applications, where measurements are typically affected by non-negligible levels of noise, and the dynamic model is subject to uncertainties. Imperfect information is considered in [6], [7], but the results are limited to a two-agents scenario. Here, we overcome these limitations by providing a solution that is robust to disturbance inputs and measurement noise, and applies to an arbitrary number of agents. Our solution is based on the computation of the (open loop) *Maximal Robust Controlled Invariant Set*, which is the largest set of states that admits an input that avoids conflicts for all positive times and *for any admissible disturbance*. This extension is nontrivial, since the Maximal Robust Controlled Invariant Set is defined over the power set of the state space, and the corresponding properties must be verified for sets of trajectories, rather than for single trajectories. The essential ingredient that allows to obtain our result is a monotonicity property of the system's flow, which allows us to define an order for the output trajectories. Membership in the Maximal Robust Controlled Invariant Set is computed by recurring to a particular scheduling algorithm [8], [9]. By this means, we can prove that our problem is NP-hard, but we can also provide an approximate solution, with polynomial complexity and provable error bound. Moreover, both exact and approximate solutions are decidable [10], that is, they can be computed in a finite number of steps. To simplify the discussion, we present our result as the solution of two separate problems, written in terms of a *state estimate*, that is, a set of states compatible with the information available at a given time.

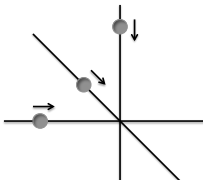

Fig. 1. Three agents must cross the intersection while avoiding collisions

1) *Verification Problem:* Given the initial state estimate of a set of n agents moving along n different paths crossing at an intersection, determine if there exists an input signal that leads all agents through the intersection avoiding collisions, for all possible disturbances and for all initial conditions in the initial state estimate.
2) *Supervisor Problem:* Design a supervisor that, given a desired input, returns the desired input unless this may cause a collision at some future time, in which case it returns a safe input.

Notice that we assume that agents move along different paths, and that all paths intersect at a common point, as in Fig. 1. We first solve the Verification Problem both exactly and approximately using an equivalent scheduling problem. These solutions are then used to design the supervisor. The paper is organized as follows. In Section 2, we define the model and the notation; in Section 3, we introduce and solve the Verification Problem; in Section 4, we introduce and

This work was supported by NSF Award # CNS 0930081 and by Progetto Rocca.

Leonardo Bruni and Alessandro Colombo are with the Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, via Ponzio 34/5, 20133, Milano, Italy. Email: leonardo.bruni@mail.polimi.it, alessandro.colombo@polimi.it.

Domitilla Del Vecchio is with the Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, USA. Email: ddv@mit.edu.

solve the Supervisor Problem; in Section 5, we present some simulation results.

II. Model and notation

Consider the system

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

where $\mathbf{x}$ is the state of n agents moving on n different paths (such as in Fig. 1), $\mathbf{x}_m$ is the measured value of $\mathbf{x}$, $\mathbf{y}$ is the vector of the positions of the agents along their paths, $\mathbf{u}$ is a control input, and $\mathbf{d}$ is a disturbance input. The measurement $\mathbf{x}_m$ is affected by an uncertainty δ. System (1) is given by the parallel composition of n different systems that describe the longitudinal dynamics of each agent:

$$\dot{x}_i = f_i(x_i, u_i, d_i),\ x_{i,m} = x_i + \delta_i,\ y_i = h_i(x_i), \tag{2}$$

with $x_i \in X_i \subseteq \mathbb{R}^r$, $\delta_i \in \Delta_i \subset \mathbb{R}^r$, $y_i \in Y_i \subset \mathbb{R}$, $u_i \in U_i \subset \mathbb{R}^s$, and $d_i \in D_i \subset \mathbb{R}^s$. Throughout the text, the symbols x_i, δ_i, y_i, u_i, and d_i will be used indifferently to denote vectors (as above) and signals, that is, functions of time. The correct interpretation will be clear from the context. The value of x_i at time $t + t_0$, starting from $x_i(t_0)$, with input signals u_i and d_i in $[t_0, t]$, is denoted $x_i(t, u_i, d_i, x_i(t_0))$. When some of the arguments are inessential, they are simply omitted. The same notation is used for y. The functional space of the input signals $u_i(t)$ is $\mathcal{U}_i$.

We assume that (1) has unique solutions and that systems (2) are monotone [11], with $\mathbb{R}_+$ as the positivity cone of y_i. This means that, given the positivity cones $K_{i,x} \subset \mathbb{R}^r$, $K_{i,u} \subset \mathbb{R}^s$, and $K_{i,d} \subset \mathbb{R}^s$, if $x_i(t_0) - x_i'(t_0) \in K_{i,x}$, $u_i(t) - u_i'(t) \in K_{i,u}$, and $d_i(t) - d_i'(t) \in K_{i,d}$ for all $t \geq t_0$, then $x_i(t) - x_i'(t) \in K_{i,x}$ and $y_i(t) - y_i'(t) \in \mathbb{R}_+$ for all $t \geq t_0$. For a vector x_i, we write $x_i \leq x_i'$ if $x_i - x_i' \in K_{i,x}$, and $x_i \in [x^a, x^b]$ if $x^a \leq x_i \leq x^b$. For a signal x_i, we write $x_i \leq x_i'$ if $x_i - x_i' \in K_{i,x}$ for all $t \geq 0$. The same notation is used for all other variables. As we have seen in (1), we assume that the state $\mathbf{x}$ is only imperfectly measured and that the initial state estimate is an interval $[x_i^a, x_i^b] \subset 2^{X_i}$ such that $\{x_i \in X_i : x_i \in [x_i^a, x_i^b]\}$. We write $[\mathbf{x}^a, \mathbf{x}^b] := [x_1^a, x_1^b] \times \ldots \times [x_n^a, x_n^b]$. We also assume that the sets Δ_i, U_i, and D_i are compact, with a unique maximum, $\delta_{i,max}$, $u_{i,max}$, and $d_{i,max}$, and minimum, $\delta_{i,min}$, $u_{i,min}$, and $d_{i,min}$, respectively. Maxima and minima are intended in the orders induced by $K_{i,x}$, $K_{i,u}$, and $K_{i,d}$, respectively. We denote $\mathbf{d}_{min}$ the vector $[d_{1,min}, \ldots, d_{n,min}]$ and $\mathbf{d}_{max}$ the vector $[d_{1,max}, \ldots, d_{n,max}]$. We assume that $\dot{y}_i$ is bounded to a strictly positive interval $[\dot{y}_{i,min}, \dot{y}_{i,max}]$ for all i. Finally, we assume that there exists a nondecreasing function β such that $\beta(0, 0, 0) = 0$ and

$$\begin{aligned}\|x_i(t, u_i, d_i, x_i(t_0)) - x_i(t, u_i, d_i', x_i'(t_0))\| \leq \\ \beta(t, \|d_i - d_i'\|, \|x_i(t_0) - x_i'(t_0)\|)\end{aligned} \tag{3}$$

for all $u_i \in \mathcal{U}_i$, $d_i, d_i' \in D_i$, and $x_i(t_0), x_i'(t_0) \in X_i$. Note that, if solutions exist for all $t \geq t_0$, this is equivalent to assuming that (2) is incrementally forward complete [12]. From now on, given an initial state estimate $[x_i^a, x_i^b]$, we use $y_i^a(t, u_i)$ to denote the position reached at time t with initial state x_i^a and disturbance input $d_{i,min}$, and $y_i^b(t, u_i)$ to denote the position reached at time t with initial state x_i^b and disturbance input $d_{i,max}$, i.e., $y_i^a(t, u_i) := y_i(t, u_i, d_{i,min}, x_i^a)$ and $y_i^b(t, u_i) := y_i(t, u_i, d_{i,max}, x_i^b)$.

III. Verification Problem

We assign to each agent an open interval (a_i, b_i), that represents the span of the intersection along the agent's path (this must account for the physical size of the agent and of the intersection). A collision occurs when two agents verify the conditions $y_i(t) \in (a_i, b_i)$ and $y_j(t) \in (a_j, b_j)$ at the same instant t. We call *Bad Set* the subset $B \subset Y \subseteq \mathbb{R}^n$ of collision points, defined as: $B := \{\mathbf{y} \in \mathbf{Y} : y_i \in (a_i, b_i) \wedge y_j \in (a_j, b_j), \text{for some } i \neq j\}$. The shape of this set changes based on the number of agents: B is open and bounded for $n = 2$ but it becomes unbounded for $n > 2$. The Verification Problem consists in determining whether there exists an input that avoids the Bad Set. In order to formalise the Verification Problem, we use the concept of Maximal Robust Controlled Invariant Set, M_R:

Definition 1: A set $S \subset X$ belongs to M_R if and only if there exists $\mathbf{u} \in \mathcal{U}$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}(t_0)) \notin B$ for all $t \geq t_0$, for all $\mathbf{x}(t_0) \in S$, and for all signals $\mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]$. Notice that, while the literature on the Maximal Robust Controlled Invariant Set [13] is based on closed loop maps, the definition above refers to open loop controls. Note also that, given an initial state estimate $[\mathbf{x}^a, \mathbf{x}^b]$, the definition amounts to having an input $\mathbf{u} \in \mathcal{U}$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}(t_0)) \notin B$ for all $t \geq t_0$, for all $\mathbf{x}(t_0) : \mathbf{x}^a \leq \mathbf{x}(t_0) \leq \mathbf{x}^b$, and for all signals $\mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]$. The approach we follow consists in mapping the Verification Problem onto an equivalent scheduling problem that we call *Problem 1*. A scheduling problem consists in assigning to a number of jobs one or more resources satisfying given requirements. In this case, the intersection represents the resource, the agents represent the job to be assigned to the resource, and the time spent by each agent in the intersection is the length of the job to be executed. We prove the equivalence rigorously. In order to achieve this, we first have to introduce some well known notions from the literature on Scheduling and Computational Complexity Theory. The standard formalism to describe a scheduling problem, introduced in [14], represents a problem by the string $\alpha|\beta|\gamma$, where α describes the resource environment (e.g. the number of resources), β defines the jobs characteristics (e.g. the release time) and γ defines the optimality criterion (e.g minimize the maximum lateness L_{max}). In the context of this paper, since we need to determine whether or not a schedule exists, we focus on decisions problems. In computational complexity theory a *decision problem* P is a problem that has a binary answer {*yes*, *no*} [9]. When P returns "*yes*" given an instance I, we say that P accepts I, denoted $I \in P$. We use the notation DEC($\alpha|\beta|\gamma, \epsilon$) to represent the decision problem that returns "*yes*" if $\alpha|\beta|\gamma$ has a solution with $\gamma \leq \epsilon$, otherwise it returns "*no*". This paper focuses on DEC($1|r_i, p_i = 1|L_{max}, 0$), defined as follows

Definition 2: Given a set of n jobs to be run on a single machine, with release times $r_i \in \mathbb{R}_+$, deadlines $d_i \in \mathbb{R}_+$ and durations $p_i = 1$, determine if there exists a schedule $\mathbf{T} = [T_1, \ldots, T_n] \in \mathbb{R}_+^n$ such that, for all $i \in \{1, \ldots, n\}$, $r_i \leq T_i \leq d_i - p_i$, and for all $i \neq j$, $T_i \geq T_j \Rightarrow T_i \geq T_j + p_j$. DEC$(1|r_i, p_i = 1|L_{max}, 0)$ has an exact $O(n^3)$-time solution, reported in [9] and implemented in [5] by the procedure POLYNOMIALTIME.

The concepts of *reducibility* and *equivalence* [15], [8] are used when comparing the complexity of different problems.

Definition 3: A decision problem $P1$ is reducible to a decision 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 $P1 \propto P2$. Two problems $P1$ and $P2$ are equivalent, denoted $P1 \simeq P2$, if $P1 \propto P2$ and $P2 \propto P1$.

A. Formalization of the Verification Problem and of Problem 1

We can now formally define the Verification Problem:

Verification Problem: Determine if $[\mathbf{x}^a, \mathbf{x}^b] \in M_R$.

An instance I of the Verification Problem is described by the sets $[\mathbf{x}^a, \mathbf{x}^b]$ and $\Theta := \{\mathbf{f}, \mathbf{h}, X, U, \mathcal{U}, D, Y, a_1, \ldots, a_n, b_1, \ldots, b_n, \mathbf{d}_{min}, \mathbf{d}_{max}\}$.

To introduce Problem 1, for each agent we define $R_i := \min\{t \geq 0 : y_i^b(t, u_{i,max}) \geq a_i\}$ and $D_i := \min\{t \geq 0 : y_i^b(t, u_{i,min}) \geq a_i\}$. These two quantities are, respectively, the minimum and maximum time at which $y_i^b(t, u_i)$ can enter the interval $[a_i, b_i]$. Notice that R_i and D_i are always well defined, since (1) has unique solutions and $\dot{y}_i \geq \dot{y}_{i,min} > 0$. Also, for each agent such that $y_i^b(t_0) < a_i$, given a real number T_i, we define $P_i(T_i) := \inf_{u_i \in \mathcal{U}_i} \{t : y_i^a(t, u_i) = b_i\}$ with the constraint $y_i^b(t, u_i) \leq a_i$ for all $t < T_i$. If the constraint cannot be satisfied, we set $P_i := \infty$; if $[y_i^a(t_0), y_i^b(t_0)] \cap (a_i, b_i) \neq \emptyset$, we define $P_i(T_i) := \{t : y_i^a(t, u_{i,max}) = b_i\}$, and if $y_i^a(t_0) \geq b_i$ we define $P_i(T_i) := 0$. $P_i(T_i)$ is the earliest time an agent can leave the intersection, if it does not to enter it before T_i. We can now define the following scheduling problem.

Problem 1: Given $[\mathbf{x}^a, \mathbf{x}^b]$, determine if there exists a schedule $\mathbf{T} = [T_1, T_2, \ldots, T_n] \in \mathbb{R}^n$ such that, for all i

$$R_i \leq T_i \leq D_i, \tag{4}$$

and for all $i \neq j$

$$T_i \geq T_j \Rightarrow T_i \geq P_j(T_j). \tag{5}$$

As for the Verification Problem, an instance I of Problem 1 is described by $[\mathbf{x}^a, \mathbf{x}^b]$ and $\Theta := \{\mathbf{f}, \mathbf{h}, X, U, \mathcal{U}, D, Y, a_1, \ldots, a_n, b_1, \ldots, b_n, \mathbf{d}_{min}, \mathbf{d}_{max}\}$.

We can now proceed to show that the Verification Problem and Problem 1 are equivalent. To prove it, we first introduce the following lemma that is a consequence of monotonicity:

Lemma 1: Given $x_i(t_0) \in [x_i^a, x_i^b]$, $y_i^a(t, u_i) \leq y_i(t, u_i, d_i, x_i(t_0)) \leq y_i^b(t, u_i)$ for all $t \geq t_0$ and for all $d_i \in [d_{i,min}, d_{i,max}]$.

Theorem 1: Verification Problem $\simeq$ Problem 1

Proof: We have to prove that

$$I \in \text{Verification Problem} \Leftrightarrow I \in \text{Problem 1}$$

1) $I \in$ Verification Problem $\Rightarrow I \in$ Problem 1
Assume that $[\mathbf{x}^a, \mathbf{x}^b]$ satisfies the constraints of the Verification Problem. This means that there exists an input that leads all the agents safely through the intersection. We call that input $\tilde{\mathbf{u}}$. The time instants at which $\mathbf{y}^b(t, \tilde{\mathbf{u}})$ crosses each of the planes $y_i = a_i$ define a vector $\mathbf{T}$, and we can set $T_i = 0$ if $y_i^b(t_0) > a_i$. Given the definition of R_i and D_i, $\mathbf{T}$ satisfies (4). Moreover, the time instants at which $\mathbf{y}^a(t, \tilde{\mathbf{u}})$ crosses each of the planes $y_i = b_i$ define a vector $\tilde{\mathbf{P}}$ and let $\tilde{P}_i = 0$ if $y_i^a(t_0) > b_i$. Since $\tilde{\mathbf{u}}$ does not cause collisions, for all $i \neq j$ with $T_i > T_j$, $y_i^b(t, \tilde{u}_i)$ enters the intersection when $y_j^a(t, \tilde{u}_j)$ has already left it. By the definitions above, this means that for all $i \neq j$ $T_i > T_j \Rightarrow T_i \geq \tilde{P}_j$. Finally, $\tilde{P}_j \geq P_j(T_j)$ since by definition $P_j(T_j)$ is the minimum time $y_j^a(t, u_j)$ can exit the intersection if $y_j^b(t, u_j)$ does not enter earlier than T_j. This implies that $\mathbf{T}$ satisfies (5).

2) $I \in$ Verification Problem $\Leftarrow I \in$ Problem 1
Assume that the schedule $\mathbf{T}$ satisfies the constraints of Problem 1 for a given $[\mathbf{x}^a, \mathbf{x}^b]$. Assume that $y_i^b(t_0) \leq a_i$ for all i and that $T_n \geq T_i$ for all $i \in \{1, \ldots, n-1\}$. To satisfy (5), $P_i(T_i)$ must be finite for all $i \in \{1, \ldots, n-1\}$. By definition of $P_i(T_i)$ there exists an input $u_i(t)$ such that $y_i^b(t, u_i) \leq a_i$ for $t < T_i$ and $y_i^a(t, u_i) = b_i$ for $t = P_i(T_i)$. Using u_i, by Lemma 1, each $y_i(t, u_i, d_i)$ enters the interval (a_i, b_i) no earlier than T_i and leaves the interval no later that $P_i(T_i)$. Moreover, (5) implies that the time intervals $(T_i, P_i(T_i))$ do not intersect. Thus agents $1, \ldots, n-1$ do not collide. Then, setting $u_n(t) = u_{n,min}$, we know that $y_n^b(t, u_n) \leq a_n$ for all $t \leq D_n$. By (4), $D_n \geq T_n$ and by (5), $T_n \geq P_i(T_i)$ for all $i \in \{1, \ldots, n-1\}$. Thus, by Lemma 1, when $y_n(t, u_n, d_n) \in [a_n, b_n]$, $y_i(t, u_i, d_i) \geq b_i$ for all $i \in \{1, \ldots, n-1\}$, hence agents $1, \ldots, n$ do not collide.
If for some systems $y_i^b(t_0) > a_i$, then $D_i = R_i = 0$ and this implies $T_i = 0$. For agents with $y_i^a(t_0) \geq a_i$, by definition of P_i, we have that $P_i(0) = 0$ if $y_i^a(t_0) \geq b_i$, otherwise $P_i(0) > 0$. Assume that agents $1, \ldots, p-1$ have $y_i^a(t_0) \geq b_i$, that $y_p^a(t_0) < b_p$ and that $y_j^b(t_0) < a_j$ for all $j \in \{p+1, \ldots, n\}$. Agents $1, \ldots, p-1$ do not collide because they have already passed the intersection. Agent p has input such that $y_p^a(P_p(T_p), u_p) = b_p$, so, by Lemma 1, $y_p(t, u_p)$ leaves the intersection no later than $P_p(T_p)$, and the agents $p+1, \ldots, n$ reach the intersection at $t > P_p(T_p)$ without collisions by the reasoning above.

■

B. Solution

The solution of Problem 1 and, as a result, of the Verification Problem can be found using Algorithm 1.

Algorithm 1 ExactSolution

for all $i \in \{1, \dots, n\}$ **do**
  given $[x_i^a, x_i^b]$ calculate R_i and D_i
for all $\pi \in \mathcal{P}$ **do**
  $T_{\pi_1} \leftarrow R_{\pi_1}$
  for $i \in \{2, \dots, n\}$ **do**
    $T_{\pi_i} \leftarrow \max(P_{\pi_{i-1}}(T_{\pi_{i-1}}), R_{\pi_i})$
  if $T_i \le D_i$ for all $i \in \{1, \dots, n\}$ **then**
    return $\{\mathbf{T}$, *yes*$\}$
return $\{\emptyset$, *no*$\}$

Given the set of initial conditions, Algorithm 1 calculates $\mathbf{R} = [R_1, \dots, R_n]$ and $\mathbf{D} = [D_1, \dots, D_n]$. The schedule is then found, if one exists, testing all the possible permutations of the n agents of the system. Since the cardinality of the search space grows factorially in the number of agents n, so does the running time of the algorithm; this is a problem if the algorithm is to be run in real time. Note that a particular subset of Problem 1, obtained for $\delta = 0$ and $\mathbf{d} = 0$ (i.e., in the absence of all disturbances), was proved in [5] to be NP-hard by reduction of a standard scheduling problem. As a consequence, Problem 1 is itself NP-hard, so even by refining Algorithm 1 we cannot expect to significantly improve the worst case performance.

To reduce the running time of the algorithm, we provide an approximate solution to Problem 1.

Lemma 2: Assume that $\mathcal{U}$ is path connected. Then, if $y_i(t_0) < a_i$, for any $T_i \in [R_i, D_i]$ there exists a $u_i \in \mathcal{U}_i$ such that $y_i(T_i, u_i, d_{i,max}) = a_i$.

Proof: By the continuity of h_i in (2) and by the continuous dependence trajectories on the input, y_i depends continuously on u_i. Since $y_i(t_0) < a_i$, $\dot{y}_i > \dot{y}_{i,min} > 0$, and solutions are unique, $\{t : y_i(t, u_i, d_{i,max}) = a_i\}$ defines a single-valued continuous map from the path connected set $\mathcal{U}_i$ to $[R_i, D_i]$. There is a path in $\mathcal{U}_i$ connecting the inputs corresponding to R_i and D_i, and the image of a continuous path under a continuous map is a continuous path covering the interval $[R_i, D_i]$. ■

From now on, we assume that $\mathcal{U}$ is path connected. Given the initial state estimate $[\mathbf{x}^a, \mathbf{x}^b]$, using (3) let

$$\gamma_i := \beta(D_i, \|d_{i,max} - d_{i,min}\|, \|x_i^b - x_i^a\|). \tag{6}$$

Then, for each i consider the set

$$S_i := \{[x_i^a, x_i^b] \in 2^{X_i} : x_i^a \le x_i^b, y_i^b = a_i, \|x_i^b - x_i^a\| \le \gamma_i\}.$$

This is the set of all the initial state estimates whose extrema x_i^a and x_i^b have distance less than γ_i, and whose upper corner x_i^b is such that $y_i^b = a_i$. We define

$$\theta_{max} := \max_{i \in \{1,\dots,n\}} \sup_{[x_i^a(t_0), x_i^b(t_0)] \in S_i} \{t : y_i^a(t, u_{i,max}) = b_i\}. \tag{7}$$

This is the minimum worst case time that the initial state estimate will need to completely traverse the interval $[a_i, b_i]$.

Lemma 3: $P_i(T_i) - T_i \le \theta_{max}$ for all $T_i \in [R_i, D_i]$.

Proof: Consider the initial state estimate $[x_i^a, x_i^b]$ at time $t = t_0$. If $y_i^a(t_0) \ge b_i$, $P_i(T_i) = T_i = 0$ and the proof is trivial, so let us assume that $y_i^a(t_0) < b_i$. Then, either $y_i^b(t_0) \ge a_i$ and $T_i \in [R_i, D_i] = [0, 0]$ or, by Lemma 2, for all $T_i \in [R_i, D_i]$ there exists a u_i such that $y_i^b(T_i, u_i) = a_i$. In both cases we can pick an input u_i such that $y_i^b(T_i, u_i) \ge a_i$, and by (6) and (3) $\|x_i^b(T_i, u_i) - x_i^a(T_i, u_i)\| \le \gamma_i$. By (7), applying the input $u_{i,max}$ for $t \ge T_i$, we have that $y_i^a(T_i + \theta_{max}, u_{i,max}) \ge b_i$. Thus, by the definition of $P_i(T_i)$, $P_i(T_i) - T_i \le \theta_{max}$ ■

We use (7) to allocate the resource of the scheduling problem – the intersection. This means that we are considering the intersection occupied for more time then is strictly needed by each agent. We are thus trading maximum traffic flow for computational speed.

We now define the approximate scheduling problem:

Problem 2: Given $[\mathbf{x}^a, \mathbf{x}^b]$, determine if there exists a schedule $\mathbf{T} = [T_1, T_2, \dots, T_n] \in \mathbb{R}^n$ such that, for all i, $R_i \le T_i \le D_i$, and for all $i \ne j$, $T_i \ge T_j \Rightarrow T_i \ge T_j + \theta_{max}$.

Any schedule that satisfies the constraints of Problem 2 also satisfies the constraints of Problem 1, since by Lemma 3 $T_j + \theta_{max} \ge P_j(T_j)$. By normalizing the data of Problem 2 to make $\theta_{max} = 1$, and then setting $R_i = r_i$, $D_i = d_i - 1$, $T_i = t_i$, Problem 2 for agents with R_i, $D_i > 0$ becomes formally equivalent to DEC$(1|r_i, p_i = 1|L_{max}, 0)$, which is solved in polynomial time by the procedure POLYNOMIALTIME in [5]. Algorithm 2 solves Problem 2 treating separately agents with $y_i^b(t_0) < a_i$ and agents with $y_i^b(t_0) \ge a_i$, for which $R_i = D_i = T_i = 0$ so that they do not contribute to the combinatorial complexity of the problem. Without loss of generality, in the algorithm we assume that $y_i^b(t_0) \ge a_i$ for $i \in \{1, \dots, m\}$, and that $y_i^b(t_0) < a_i$ for $i > m$.

Algorithm 2 ApproximateSolution

for all $i \in \{1, \dots, n\}$ **do**
  given $[x_i^a, x_i^b]$ calculate R_i e D_i
if $[y_i^a(t_0), y_i^b(t_0)] \cap (a_i, b_i) \ne \emptyset$ for two different $i \in 1, \dots, m$ **then**
  return $\{\emptyset$, *no*$\}$
for all $i \in \{1, \dots, m\}$ **do**
  $T_i \leftarrow 0$
$R_{bound} \leftarrow \max\{P_1(t_0), P_2(t_0), \dots\}$
for all $i \in \{m+1, \dots, n\}$ **do**
  $R_i \leftarrow \max(R_i, R_{bound})$
calculate θ_{max}
$\mathbf{r} = (R_{m+1}/\theta_{max}, \dots, R_n/\theta_{max})$
$\mathbf{d} = (D_{m+1}/\theta_{max} + 1, \dots, D_n/\theta_{max} + 1)$
$\{T_{m+1}, \dots, T_n$, *answer*$\}$=POLYNOMIALTIME$(\mathbf{r}, \mathbf{d})$
for i from $m+1$ to n **do**
  $T_i \leftarrow T_i * \theta_{max}$
return $\{\mathbf{T}$, *answer*$\}$

We now need to understand how good the solution is. To do this, we want to find an upper bound to the overestimation of the Bad Set due to Algorithm 2. This is done in Theorem 2 using the following intermediate result.

Lemma 4: For a given initial state estimate $[\mathbf{x}^a, \mathbf{x}^b]$, take an arbitrary $\mathbf{u} \in \mathcal{U}$, and define a schedule $\mathbf{T}$ as $T_i = \{t : y_i^b(t, u_i) = a_i\}$ for all i such that $y_i^b(t_0) < a_i$, and $T_i = 0$ for all other i. Assume that, for some i and j, $y_i^b(t_0) < a_i$, $y_j^b(t_0) \leq a_j$, $T_i \geq T_j$, and $T_i - T_j \leq \theta_{max}$, that is, two jobs are scheduled within θ_{max} of each other. Then $\inf_{\mathbf{x}(t_0)\in[\mathbf{x}^a,\mathbf{x}^b],\mathbf{d}\in[\mathbf{d}_{min},\mathbf{d}_{max}]} \inf_{t\geq t_0, b\in B} ||\mathbf{y}(t,\mathbf{u},\mathbf{d},\mathbf{x}(t_0)) - b||_\infty \leq \max_{i\in\{1,\dots,n\}} \frac{\dot{y}_{i,max}}{2}\left(\theta_{max} - \frac{b_i - a_i}{\dot{y}_{i,max}}\right)$.

Proof: To prove the lemma, it is sufficient to show that $\inf_{t\geq t_0, b\in B} ||\mathbf{y}^b(t,\mathbf{u}) - b||_\infty \leq \max_{i\in\{1,\dots,n\}} \frac{\dot{y}_{i,max}}{2}\left(\theta_{max} - \frac{b_i - a_i}{\dot{y}_{i,max}}\right)$. First, notice that the right hand side of the inequality is always positive, since $\dot{y}_{i,max}\theta_{max} \geq b_i - a_i$. Let $\dot{y}_{max} := \max_i \dot{y}_{i,max}$, and $\dot{y}_{min} := \min_i \dot{y}_{i,min}$. Any trajectory that satisfies $y_i(T_i) = a_i$ and $y_j(T_j) = a_j$ must satisfy the three following sets of inequalities: $\forall t \leq T_j : \left\{y_j(t) \leq (t - T_j)\dot{y}_{min} + a_j, y_i(t) \geq (t - T_i)\dot{x}_{max} + a_i, T_j - t \leq \frac{a_j - y_j(t)}{\dot{y}_{min}}\right\}$; $\forall t \in [T_j, T_i] : \left\{y_j(t) \leq (t - T_j)\dot{y}_{max} + a_j, y_i(t) \geq (t - T_i)\dot{y}_{max} + a_i\right\}$; and $\forall t \geq T_i : \left\{y_j(t) \geq (t - T_j)\dot{y}_{min} + a_j, y_i(t) \leq (t - T_i)\dot{y}_{max} + a_i, t - T_i \geq \frac{y_i(t) - a_i}{\dot{y}_{max}}\right\}$. By removing the dependence on t of the above inequalities we obtain the equation of the boundaries of the region containing all such trajectories in the (y_i, y_j) plane. We conclude that these trajectories lie between the curve of equations

$$\begin{cases} y_j = (y_i - a_i)\frac{\dot{y}_{min}}{\dot{y}_{max}} + a_j + (T_i - T_j)\dot{y}_{min} \text{ if } y_i \leq a_i, y_j \leq a_j \\ y_j = y_i - a_i + a_j + (T_i - T_j)\dot{y}_{max} \text{ if } y_i \leq a_i, y_j \geq a_j \\ y_j = (y_i - a_i)\frac{\dot{y}_{max}}{\dot{y}_{min}} + a_j + (T_i - T_j)\dot{y}_{max} \text{ if } y_i \geq a_i, y_j \geq a_j \end{cases}$$

and the curve of equations

$$\begin{cases} y_j = (y_i - a_i)\frac{\dot{y}_{max}}{\dot{y}_{min}} + a_j + (T_i - T_j)\dot{y}_{max} \text{ if } y_i \leq a_i, y_j \leq a_j \\ y_j = y_i - a_i + a_j + (T_i - T_j)\dot{y}_{min} \text{ if } y_i \leq a_i, y_j \geq a_j \\ y_j = (y_i - a_i)\frac{\dot{y}_{min}}{\dot{y}_{max}} + a_j + (T_i - T_j)\dot{y}_{min} \text{ if } y_i \geq a_i, y_j \geq a_j. \end{cases}$$

Now, since $(T_i - T_j) \leq \theta_{max}$, by substituting this in the two above sets of equations we obtain an upper and lower bound of the region. Let S be the region enclosed by these two bounds. The bound in the statement then follows from taking $\inf_{t\geq 0, b\in B, \mathbf{y}\in S} \|\mathbf{y} - b\|_\infty$. ∎

Theorem 2: If, for a given $[\mathbf{x}^a, \mathbf{x}^b]$, Algorithm 2 returns "*no*", then

$$\sup_{\mathbf{u}\in\mathcal{U}} \inf_{\mathbf{x}(t_0)\in[\mathbf{x}^a,\mathbf{x}^b],\mathbf{d}\in[\mathbf{d}_{min},\mathbf{d}_{max}]} \inf_{t\geq t_0, b\in B} ||\mathbf{y}(t,\mathbf{x}(t_0),\mathbf{u},\mathbf{d}) - b||_\infty \leq \max_{i\in\{1,\dots,n\}} \frac{\dot{y}_{i,max}}{2}\left(\theta_{max} - \frac{b_i - a_i}{\dot{y}_{i,max}}\right). \tag{8}$$

Proof: Algorithm 2 returns "*no*" if $[y_i^a(t_0), y_i^b(t_0)] \cap [a_i, b_i) \neq \emptyset$ for two different i, or if POLYNOMIALTIME returns "*no*". In the first case the left hand side of (8) is equal to 0 and (8) is verified. In the second case, if POLYNOMIALTIME returns "*no*" then, for any schedule $\mathbf{T}$ with $T_i \in [R_i, D_i]$ for all i, there exist i and j with $y_i^b(t_0) < a_i$, $y_j^b(t_0) < a_j$, $T_j \leq T_i$, such that $T_i - T_j < \theta_{max}$. This is a consequence of the fact that POLYNOMIALTIME solves DEC$(1|r_i, p_i = 1|L_{max}, 0)$ exactly. By the reasoning above, for any $\mathbf{u} \in U$, the schedule $\mathbf{T}$ defined by $T_i = \{t : y_i^b(t, u_i, d_{i,max}) = a_i\}$ if $y_i^b(t_0) < a_i$, $T_i = 0$ otherwise, has $T_i \in [R_i, D_i]$ for all i, and satisfies the hypotheses of Lemma 4. This completes the proof. ∎

According to Theorem 2, if Algorithm 2 cannot find a schedule that satisfies Problem 2, for all $\mathbf{u} \in \mathcal{U}$ there exists at least one $\mathbf{x}(t_0) \in [\mathbf{x}^a, \mathbf{x}^b]$ and $\mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]$ such that $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}(t_0))$ intersects the Extended Bad Set, defined as follows

$$\hat{B} := \left\{\mathbf{y} : \inf_{b\in B} ||\mathbf{y} - b||_\infty \leq \max_{i\in\{1,\dots,n\}} \frac{\dot{y}_{i,max}}{2}\left(\theta_{max} - \frac{b_i - a_i}{\dot{y}_{i,max}}\right)\right\}. \tag{9}$$

IV. SUPERVISOR PROBLEM

We now formally introduce and solve the Supervisor Problem. Our goal is to design a supervisor that keeps the state estimate inside the Maximal Robust Controlled Invariant Set using either the input chosen by the drivers, called the *desired input*, if this does not cause a collision at some future time, or a safe input.

A. Formalization

We call $\mathbf{v}_k$ the desired input of the system at time $k\tau$. Then we consider the signal $\mathbf{u}_k$, defined over the interval $[k\tau, (k+1)\tau]$, that is equal to $\mathbf{v}_k$ in the whole interval. We call $\mathbf{u}_k^\infty(t)$ the signal defined in $t \in [k\tau, +\infty)$ so that, in the interval $[k\tau, (k+1)\tau]$, $\mathbf{u}_k^\infty = \mathbf{u}_k$. Consider also the signal $\mathbf{u}_{k,s}^\infty(t)$ (if it exists) such that $\mathbf{y}^a(t, \mathbf{u}_{k,s}^\infty) \notin B$ and $\mathbf{y}^b(t, \mathbf{u}_{k,s}^\infty) \notin B$ for all $t \geq k\tau$, and let $\mathbf{u}_{k,s}$ be the same signal restricted to the interval $[k\tau, (k+1)\tau]$. Finally, we design the one-step ahead predictor of the state of system (1)

$$\begin{cases} \mathbf{x}_{pred}^{max}(\tau, \mathbf{u}) := \mathbf{x}(\tau, \mathbf{u}, \mathbf{d}_{max}, \mathbf{x}^b(k\tau)) \\ \mathbf{x}_{pred}^{min}(\tau, \mathbf{u}) := \mathbf{x}(\tau, \mathbf{u}, \mathbf{d}_{min}, \mathbf{x}^a(k\tau)), \end{cases} \tag{10}$$

and the measurement error corrector that defines the initial state estimate at time $(k+1)\tau$

$$\begin{cases} \mathbf{x}^a((k+1)\tau) = \max(\mathbf{x}_{pred}^{min}(\tau, \mathbf{u}), \mathbf{x}_m((k+1)\tau) + \delta_{min}) \\ \mathbf{x}^b((k+1)\tau) = \min(\mathbf{x}_{pred}^{max}(\tau, \mathbf{u}), \mathbf{x}_m((k+1)\tau) + \delta_{max}). \end{cases} \tag{11}$$

From here on, we denote by $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$ the initial state estimate obtained at the k-th iteration of (10) and (11).

Supervisor problem: Given $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$, design a supervisor $s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k) : 2^{\mathbb{R}^r} \times \mathbb{R}^s \to \mathbb{R}^s$ for system (1) such that (i)

$$s = \begin{cases} \mathbf{u}_k & \text{if } \exists \mathbf{u}_k^\infty \in \mathcal{U} : \\ & [\mathbf{y}^a(t, \mathbf{u}_k), \mathbf{y}^b(t, \mathbf{u}_k)] \cap B = \emptyset \\ & \text{for all } t \geq 0 \\ \mathbf{u}_{k,s} & \text{otherwise} \end{cases}$$

and such that (ii) it is *nonblocking*: if $\mathbf{u} = s[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \nu_k) \neq \emptyset$ then for any $\mathbf{v}_{k+1}$, $k > 0$, $s([\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \mathbf{v}_{k+1}) \neq \emptyset$.

B. Solution

Using the solution of the Verification Problem we can solve the Supervisor Problem. We define, for all the agents with $y_i^b(t_0) < a_i$, $\sigma_i([x_i^a, x_i^b], T_i) := \arg \inf_{u_i \in \mathcal{U}_i} \{t : y_i^a(t, u_i) = b_i\}$ with constraint $y_i^b(t, u_i) \leq a_i$ for $t < T_i$. If $[y_i^a(t_0), y_i^b(t_0)] \cap (a_i, b_i) \neq \emptyset$, we define $\sigma_i([x_i^a, x_i^b], T_i) := u_{i,max}$; otherwise, we define $\sigma_i([x_i^a, x_i^b], T_i) := 0$. This is the input $u_i(t)$ that allows agent i to exit the intersection no later than $t = P_i(T_i)$ not entering before T_i.

Algorithm 3 solves the Supervisor Problem using the procedure ExactSolution in Algorithm 1.

Algorithm 3 SupervisorExact

$\mathbf{u}_k(t) \leftarrow \mathbf{v}_k \quad \forall t \in [0, \tau]$
{$\mathbf{T}$, *answer*}$\leftarrow$
ExactSolution($[\mathbf{x}_{pred}^{min}(\tau, \mathbf{u}_k), \mathbf{x}_{pred}^{max}(\tau, \mathbf{u}_k)]$)
if *answer* = *yes* and $[\mathbf{y}^a(t, \mathbf{u}_k), \mathbf{y}^b(t, \mathbf{u}_k)] \cap B = \emptyset$ for all $t \in [0, \tau]$ **then**
  $\mathbf{u}_{s,k+1}^{\infty} \leftarrow \sigma([\mathbf{x}_{pred}^{min}(\tau, \mathbf{u}_k), \mathbf{x}_{pred}^{max}(\tau, \mathbf{u}_k)], \mathbf{T})$
  $\mathbf{u}_{s,k+1} \leftarrow \mathbf{u}_{s,k+1}^{\infty}$ in $[0, \tau]$
  return $\mathbf{u}_k$
else
  {$\mathbf{T}$, *answer*}$\leftarrow$
  ExactSolution($[\mathbf{x}_{pred}^{min}(\tau, \mathbf{u}_{s,k}), \mathbf{x}_{pred}^{max}(\tau, \mathbf{u}_{s,k})]$)
  $\mathbf{u}_{s,k+1}^{\infty} \leftarrow \sigma([\mathbf{x}_{pred}^{min}(\tau, \mathbf{u}_{s,k}), \mathbf{x}_{pred}^{max}(\tau, \mathbf{u}_{s,k})], \mathbf{T})$
  $\mathbf{u}_{s,k+1} \leftarrow \mathbf{u}_{s,k+1}^{\infty}$ in $[0, \tau]$
  return $\mathbf{u}_{s,k}$

We now prove that this algorithm solves the supervisory problem.

Lemma 5: If there exists a schedule $\mathbf{T}$ that satisfies the constraints of Problem 1 given $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$, then $\sigma([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{T}) \neq \emptyset$.

Proof: The proof proceeds as the second part of the proof of Theorem 1. ■

Lemma 6: If there exists a schedule $\mathbf{T}$ that satisfies the constraints of Problem 1 given $[\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)]$, defining $\mathbf{u} := \sigma([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{T}) \neq \emptyset$, then there exists also a schedule that satisfies the constraints of Problem 1 given the new initial state estimate $[\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)]$.

Proof: The input $\mathbf{u}$ is well defined for Lemma 5. Let $\tilde{\mathbf{u}}$ be $\mathbf{u}$ restricted to the interval $[(k+1)\tau, \infty)$ and call $\tilde{\mathbf{y}}^a(t, \tilde{\mathbf{u}}) := \mathbf{y}^a(t, \tilde{\mathbf{u}}, \mathbf{d}_{min}, \mathbf{x}^a((k+1)\tau))$ and $\tilde{\mathbf{y}}^b(t, \tilde{\mathbf{u}}) := \mathbf{y}^b(t, \tilde{\mathbf{u}}, \mathbf{d}_{max}, \mathbf{x}^b((k+1)\tau))$. By Lemma 1, and since by (10) and (11) $[\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)] \subseteq [\mathbf{x}(t, \tilde{\mathbf{u}}, \mathbf{d}_{min}, \mathbf{x}^a(k\tau)), \mathbf{x}(t, \tilde{\mathbf{u}}, \mathbf{d}_{max}, \mathbf{x}^b(k\tau))]$, if $[\mathbf{y}^a(t, \mathbf{u}), \mathbf{y}^b(t, \mathbf{u})] \cap B = \emptyset$ then $[\tilde{\mathbf{y}}^a(t, \tilde{\mathbf{u}}), \tilde{\mathbf{y}}^b(t, \tilde{\mathbf{u}})] \cap B = \emptyset$; in other words the new set of possible trajectories defined from time $(k+1)\tau$ applying $\tilde{\mathbf{u}}$ does not enter in the Bad Set, therefore $\{[\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \Theta\} \in$ Verification Problem. Since the Verification Problem is equivalent to Problem 1, $\{[\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \Theta\} \in$ Problem 1. ■

Theorem 3: Assume that $s([\mathbf{x}^a(0), \mathbf{x}^b(0)], \mathbf{v}_0) \neq \emptyset$. Then, the supervisor $s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$ solves the Supervisor Problem.

Proof: To prove (i), Algorithm 3 returns $\mathbf{u}_k$ unless a new schedule is not found. If this happens, the instance $I \notin$ Problem 1 and by Theorem 1, $I \notin$ Verification Problem. Thus, by definition of the Verification Problem, there is no $\bar{\mathbf{u}}_k^{\infty}$ that is safe.
To prove (ii), we proceed by induction: assuming that $s([\mathbf{x}^a(0), \mathbf{x}^b(0)], \mathbf{v}_0) \neq \emptyset$, we have to prove that if we apply $\mathbf{u}_{out} = s([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k) \neq \emptyset$ then $s([\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \mathbf{v}_{k+1}) \neq \emptyset$. First of all, notice that $s([\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \mathbf{v}_{k+1}) \neq \emptyset$ unless $\mathbf{u}_{s,k+1} \neq \emptyset$. The argument of the procedure ExactSolution in Algorithm 3 is $[\mathbf{x}_{pred}^{min}((k+1)\tau, \mathbf{u}), \mathbf{x}_{pred}^{max}((k+1)\tau, \mathbf{u})]$ that is reached either with an input $\mathbf{u} = \mathbf{u}_k$ or $\mathbf{u} = \mathbf{u}_{s,k}$. In the first case, by Lemma 5 we are sure that $\sigma([\mathbf{x}_{pred}^{min}((k+1)\tau, \mathbf{u}), \mathbf{x}_{pred}^{max}((k+1)\tau, \mathbf{u})], \mathbf{v}_{k+1})$ is not empty. In the second case, by Lemma 6, a schedule exists and by Lemma 5, $\sigma([\mathbf{x}_{pred}^{min}((k+1)\tau, \mathbf{u}), \mathbf{x}_{pred}^{max}((k+1)\tau, \mathbf{u})], \mathbf{v}_{k+1})$ is not empty. Since, by (10) and (11), $[\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)] \in [\mathbf{x}_{pred}^{min}((k+1)\tau, \mathbf{u}), \mathbf{x}_{pred}^{max}((k+1)\tau, \mathbf{u})]$, in both cases $\sigma([\mathbf{x}^a((k+1)\tau), \mathbf{x}^b((k+1)\tau)], \mathbf{v}_{k+1})$ is not empty. ■

Due to its exponential complexity, Algorithm 3 cannot be used in the presence of a large number of agents. We could use Algorithm 2 instead of the procedure ExactSolution in Algorithm 3. This, however, would lead to a blocking algorithm. To ensure nonblockingness, we modify Algorithm 3 as detailed in Algorithm 4. To guarantee its nonblockingness, the following strategy has been adopted: if ApproximateSolution($[\mathbf{x}_{pred}^{min}((k+1)\tau, \mathbf{u}_{s,k}), \mathbf{x}_{pred}^{max}((k+1)\tau, \mathbf{u}_{s,k})]$) returns "*no*", the system keeps using the last safe input calculated at the previous step until a new schedule that satisfies the constraints of Problem 2 can be found. Note that in this case Algorithm 4 is working in open-loop until it finds a feasible schedule, since the procedure ApproximateSolution does not return any value. We now prove that Algorithm 4 is no more restrictive than Algorithm 3 defined using the Extended Bad Set in (9).

Theorem 4: Consider the Extended Bad Set defined in (9). Call $\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$ the supervisor defined in the Supervisor Problem substituting $\hat{B}$ to B, and call $s_{approx}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$ the supervisor defined by Algorithm 4. Then $s_{approx}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$ is no more restrictive than $\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$, that is, if $s_{approx}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k) = \mathbf{u}_{k,s}$ then $\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k) = \mathbf{u}_{k,s}$.

Proof: ExactSolution($[\mathbf{x}_{pred}^{min}((k+1)\tau), \mathbf{x}_{pred}^{max}((k+1)\tau)]$) in Algorithm 3 defined as in Theorem 4 returns "*yes*" if there exists an input $\mathbf{u}$ that keeps $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}((k+1)\tau))$ outside $\hat{B}$ for all $\mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]$ and for all $\mathbf{x}((k+1)\tau) \in [\mathbf{x}_{pred}^{min}((k+1)\tau), \mathbf{x}_{pred}^{max}((k+1)\tau)]$. By Theorem 2 and by (9), ApproximateSolution($[\mathbf{x}_{pred}^{min}((k+1)\tau), \mathbf{x}_{pred}^{max}((k+1)\tau)]$) returns "*yes*" if there is an input $\mathbf{u}$ that keeps $\mathbf{y}(t, \mathbf{u}, \mathbf{d}, \mathbf{x}((k+1)\tau))$ outside $\hat{B}$. So $s_{approx}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$ is no more restrictive than $\hat{s}([\mathbf{x}^a(k\tau), \mathbf{x}^b(k\tau)], \mathbf{v}_k)$. ■

Algorithm 4 SupervisorApproximate

$\mathbf{u}_k(t) \leftarrow \mathbf{v}_k \quad \forall t \in [0,\tau]$
$\{\mathbf{T}, \textit{answer}\} \leftarrow$
ApproximateSolution($[\mathbf{x}^{min}_{pred}(\tau,\mathbf{u}_k), \mathbf{x}^{max}_{pred}(\tau,\mathbf{u}_k)]$)
if *answer* = *yes* and $[\mathbf{y}^a(t,\mathbf{u}_k), \mathbf{y}^b(t,\mathbf{u}_k)] \cap B = \emptyset$ for all $t \in [0,\tau]$ **then**
  $\mathbf{u}^{\infty}_{s,k+1} \leftarrow \sigma([\mathbf{x}^{min}_{pred}(\tau,\mathbf{u}_k), \mathbf{x}^{max}_{pred}(\tau,\mathbf{u}_k)], \mathbf{T})$
  $\mathbf{u}_{s,k+1} \leftarrow \mathbf{u}^{\infty}_{s,k+1}$ in $[0,\tau]$
  return $\mathbf{u}_k$
else
  $\{\mathbf{T}, \textit{answer}\} \leftarrow$
  ApproximateSolution($[\mathbf{x}^{min}_{pred}(\tau,\mathbf{u}_{s,k}), \mathbf{x}^{max}_{pred}(\tau,\mathbf{u}_{s,k})]$)
  if *answer* = *yes* **then**
    $\mathbf{u}^{\infty}_{s,k+1} \leftarrow \sigma([\mathbf{x}^{min}_{pred}(\tau,\mathbf{u}_{s,k}), \mathbf{x}^{max}_{pred}(\tau,\mathbf{u}_{s,k})], \mathbf{T})$
    $\mathbf{u}_{s,k+1} \leftarrow \mathbf{u}^{\infty}_{s,k+1}$ in $[0,\tau]$
    return $\mathbf{u}_{s,k}$
  else
    $\mathbf{u}^{\infty}_{s,k+1} \leftarrow \mathbf{u}^{\infty}_{s,k}$ in $[\tau,+\infty)$
    $\mathbf{u}_{s,k+1} \leftarrow \mathbf{u}^{\infty}_{s,k+1}$ in $[0,\tau]$
    return $\mathbf{u}_{s,k}$

V. Numerical simulations

Algorithms 3 and 4 have been tested numerically on the following piecewise linear system. Assume that all the agents are described by the same double integrator modeling the longitudinal dynamics of a set of cars

$$\begin{cases} \dot{x}_{1,i}(t) = x_{2,i}(t) \\ x_{1,i,m}(t) = x_{1,i}(t) + \delta_1(t) \\ \dot{x}_{2,i}(t) = \alpha(u_i(t) \pm d_i(t)) \\ x_{2,i,m}(t) = x_{2,i}(t) + \delta_2(t) \\ y_i(t) = x_{1,i}(t), \end{cases}$$

where the acceleration of agent i depends on the desired input $u_i(t)$ and on the disturbance $d_i(t)$. d_i accounts for the disturbance and the rolling friction known with uncertainty, while

$$\alpha = \begin{cases} 0 \text{ if } x_2 = \dot{y}_{min} \wedge u < 0 \vee x_2 = \dot{y}_{max} \wedge u > 0 \\ 1 \text{ otherwise} \end{cases}$$

represents the speed saturation. The measurements $x_{1,i,m}(t)$ and $x_{2,i,m}(t)$ are both affected by uncertainties. We use for our simulation $u_i(t) \in [-2,1]\ m/s^2$, $\dot{y}(t) \in [1.39, 13.9]\ m/s$, $d(t) \in [-0.65, 0.15]\ m/s^2$, $\delta_1(t) \in [-3,3]\ m$, and $\delta_2(t) \in [-1,1]\ m/s$. The intersection is placed in $a = 90\ m$ and $b = 100\ m$ for all agents. We also assume that all the drivers want to go as fast as possible and that the supervisor takes its decision every 0.1 seconds.

Consider the case of 3 agents solved using Algorithm 3. The initial conditions, $\mathbf{x}_1 = [0,0,0]$ and $\mathbf{x}_2 = [13.9, 13.9, 13.9]$, are chosen so that, without the supervisor, there will be a collision as they go at the same constant speed. Fig. 2 shows the values at each time step of the real position (in blue and red) and the position estimate (in grey). Black lines define the intersection.

The safe input maintains the state estimate of the system always inside the Maximal Robust Controlled Invariant Set. Thus the intersection between the position estimate of the agents and Bad Set is always empty. Fig. 3 shows the Bad Set (in yellow) and the position estimate (in blue and red) in the space y_1, y_2, y_3.

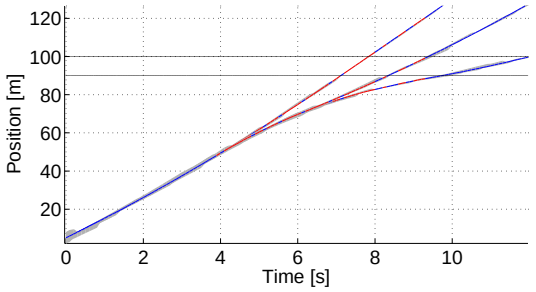

Fig. 2. The real position (union of blue and red) and the position estimate (in grey) of 3 agents over time. Blue lines indicate that the desired inputs are used, red lines that the safe inputs are used.

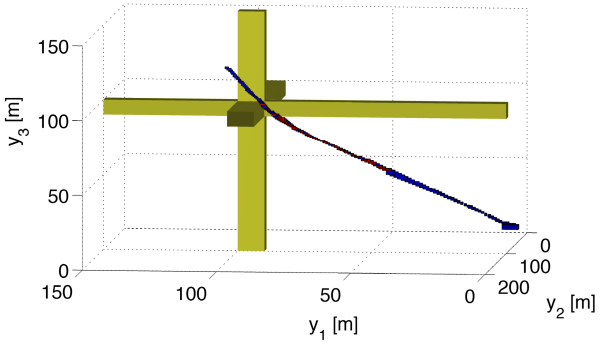

Fig. 3. The position estimate with 3 agents at every time step (blue and red cubes) and the Bad Set (in yellow) in the space y_1, y_2, y_3. When the safe input is applied, the position estimate is depicted in red, otherwise in blue.

We tested Algorithm 4 for a system composed of fifteen agents, using the same parameters as in the simulation above except for the position of the intersection which is placed in $a = 390\ m$ and $b = 400\ m$ and the initial conditions $\mathbf{x}_1 = [130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0, 0]$ and $\mathbf{x}_2 = [5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]$. Fig. 4 shows the real position (in blue and red) and the position estimate (in grey) of the agents over time. Black lines define the intersection. We define $\beta := \|x^b_i(t_0) - x^a_i(t_0)\|(1+t) + \frac{\|d_{i,max} - d_{i,min}\|t^2}{2}$. In this case, $\theta_{max} \leq 13\ s$. The agents are scheduled correctly: there are no collisions in the system, but the intersection remains free for a longer time than would be needed with Algorithm 3.

VI. Conclusion

We have addressed a class of conflict resolution problems with imperfect state information and input uncertainties. By means of an equivalence relation with a scheduling problem, we have devised exact and approximate solution algorithms, and we have used these algorithms to design a robust supervisor. The scheduling approach was initially proposed in [5] where, however, results were limited to the case of perfect state information and absence of any

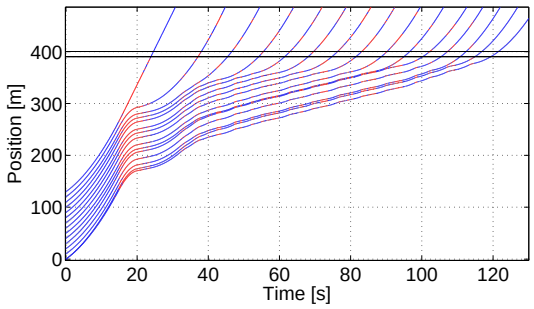

Fig. 4. The real positions (union of blue and red) and the position estimate (in grey) of 15 agents over time. Blue lines indicate that the desired inputs are used, red lines that the safe inputs are used. All the agents are scheduled using the fixed time θ_{max}.

disturbance. These limitations have been removed. Our supervisor requires to check the membership of a state estimate, consisting of a hypercube of all possible current states, in the Maximal Robust Controlled Invariant Set [13]. In the general case, this kind of problem is known to be semi-decidable [10]. However, our results show that for the class of systems considered here the problem is decidable, since the Algorithms 3 and 4 are guaranteed to terminate in a finite number of steps. Notice that our definition of the Maximal Robust Controlled Invariant Set is based on an open loop control for $t \geq 0$. We could mitigate the effects of the disturbances by using a feedback control, defining the input as a map of the current state estimate. However, integrating such a feedback control in our approach is not trivial for the general class of systems that we consider here. Extensions in this direction are currently being investigated. The present work is limited to the case of agents travelling on different paths and a single intersection. Most reasonable applications of the results, for instance to collision avoidance at road traffic intersections, require algorithms that can handle multiple agents on each path and paths intersecting at multiple points. While there are algorithms in the literature that can address these issues [16], [17], [18], [19], [20], [21], [22], the scheduling approach proposed here seems so far to be the only one to provide a robust and least restrictive solution with provable error bound from the optimum. Moreover, the approximate solution scales polynomially with the number of agents involved, and can thus be used to control a large number of agents. Extensions of the scheduling approach to address multiple agents on each path, and complex path topologies, are currently being investigated.

REFERENCES

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

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

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

[4] R. Verma and D. Del Vecchio, "Semiautonomous multivehicle safety: A hybrid control approach," *IEEE Robotics & Automation Magazine*, vol. 18, pp. 44–54, 2011.

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

[6] D. D. Vecchio, M. Malisoff, and R. Verma, "A separation principle for a class of hybrid automata on a partial order," in *American Control Conference*, 2009.

[7] M. Hafner and D. Del Vecchio, "Computation of safety control for uncertain piecewise continuous systems on a partial order." in *Coonference on Decision and Control*, 2009.

[8] 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.

[9] 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.

[10] R. Vidal, S. Schaffert, O. Shakernia., J. Lygeros, and S. Sastry, "Decidable and semi-decidable controller synthesis for classes of discrete time hybrid systems," in *40th IEEE Conference on Decision and Control*, 2001.

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

[12] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Autom. Control*, vol. 57, pp. 1804–1809, 2012.

[13] H. Lin and P. J. Antsaklis, "Robust controlled invariant set for a class of uncertain hybrid systems," in *Conference on Decision and Control*, 2002.

[14] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of discrete mathematics*, vol. 4, pp. 287–326, 1979.

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

[16] A. Colombo and D. Del Vecchio, "Supervisory control of differentially flat systems based on abstraction," in *50th IEEE Conference on Decision and Control*, 2011.

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

[18] T.-C. Au, C.-L. Fok, S. Vishwanath, C. Julien, and P. Stone, "Evasion planning for autonomous vehicles at intersections," in *IEEE/RSJ International conference on Intelligent Robots and Systems*, 2012.

[19] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Trans. Veh. Technol.*, vol. 60, pp. 804–818, 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] A. Colombo and A. Girard, "An approximate abstraction approach to safety control of differentially flat systems," in *European Control Conference*, 2013.