Symbolic Reach-Avoid Control of Multi-Agent Systems

ABSTRACT

We consider the decentralized controller synthesis problem for multi-agent systems with global reach-avoid specifications. Each agent is modeled as a nonlinear dynamical system with disturbance. The objective is to synthesize *local* feedback controllers that guarantee that the overall multi-agent system meets the global specification under the influence of disturbances. Existing techniques based on planning or trajectory optimization usually ignore the effects of disturbance and produce open-loop *nominal* trajectories which may not suffice in the presence of disturbances. Techniques based on formal synthesis that guarantee satisfaction of temporal specifications do not scale as the number of agents increase.

We address these limitations by proposing a two-level solution approach that combines fast global nominal trajectory generation and local application of formal synthesis. At the top level, we ignore the effect of disturbances and obtain a joint open-loop plan for the system using a fast trajectory optimizer. At the lower level, we use abstraction-based controller design to synthesize a set of decentralized feedback controllers that track the high level plan against worst-case disturbances, thus ensuring satisfaction of the global specification. We demonstrate the effectiveness of our approach on several multi-robot examples using two particular classes of control specifications. In the first type, we assume that the robots need to fulfill their own reach-avoid tasks while avoiding collision with the other robots. In the second type, we require the robots to fulfill reach-avoid tasks while maintaining certain formation constraints. The experiments show that our approach produces formally guaranteed feedback controllers while scaling to many robots. In contrast, nominal open loop controllers do not guarantee the satisfaction of the specification, and the exponentially large memory usage of the global formal approach prevents synthesis of a controller.

1 INTRODUCTION

We consider the *decentralized* feedback controller synthesis problem for *multi-agent*, nonlinear systems against temporal *reach-avoid* specifications. By multi-agent, we mean that the systems under study are composed of a number of concurrently executing components. Each component is modeled as a possibly nonlinear dynamical system that evolves under the influence of a control as well as an environmental disturbance. Our specifications require that the global state of the system eventually reaches a target while avoiding certain bad states along the way. While the dynamics of each component is independent of the others, the overall trajectories are coupled by the global specification. Decentralized means that we require a solution in which each component has a local feedback controller that sees only the local state, but the combination of all the closed loops satisfy the global specification. Above all, our goal is to ensure the resulting controllers are *provably correct* against the worst-case model of disturbances.

Such multi-agent control problems are ubiquitous in the domain of robotics, where a number of (possibly heterogeneous) mobile robots move concurrently in a shared workspace. A global specification can ask, for example, that a set of robots are able to reach certain locations while avoiding collisions among themselves or with obstacles in the environment, or that a set of drones fly in formation while reaching a target. Indeed, automatic generation of decentralized controllers is a classical problem in robotics, artificial intelligence, and control theory, and there is an enormous literature on the subject—too many to enumerate—across these disciplines.

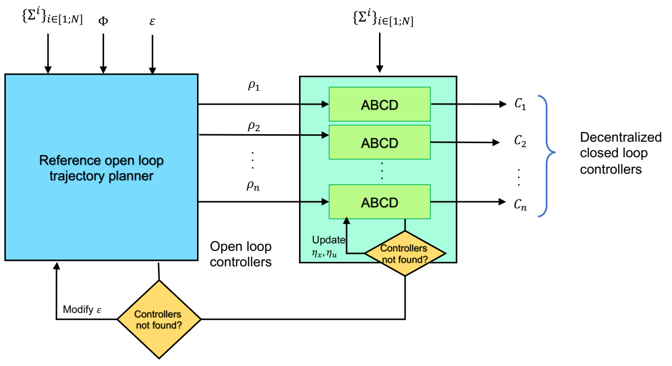


Figure 1: Overall algorithm

Despite the large body of research, few techniques today can handle all our desiderata. Multi-agent planning algorithms, such as (hybrid variants of) A* search, scale to large systems but typically either disregard or simplify the underlying dynamics and work with geometric or discrete models, or disregard the effect of disturbances or nonlinear dynamics. Most planning and trajectory optimization techniques handle the *nominal* dynamics, i.e., the dynamics free of disturbances, and construct *open-loop* controllers. However, the open-loop behaviors do not guarantee satisfaction of the specifications in the presence of disturbances. On the other hand, correct-by-construction controller synthesis techniques from control theory, such as abstraction-based control or Hamilton-Jacobi techniques, handle precise models of nonlinear dynamics and the effects of disturbance, but are difficult to scale beyond tens of dimensions.

In this paper, we provide a simple but effective *combined* approach, where a global planning approach is used for nominal trajectory generation and a local correct-by-construction feedback controller synthesis approach for guaranteed adherence to specification for each component in the presence of disturbances. Fig. 1 shows the overall algorithm. In the first step, given a set of control systems, one for each component, and a reach-avoid specification on the global state space, we use a trajectory planner to find a nominal open-loop controller for the global system. The trajectory planner ignores the effect of disturbances, but takes a *robustness parameter* ε. The role of the robustness parameter is to ensure the specification is robustly satisfied: any trajectory within an ε-tube of the open loop trajectory also satisfies the specification.

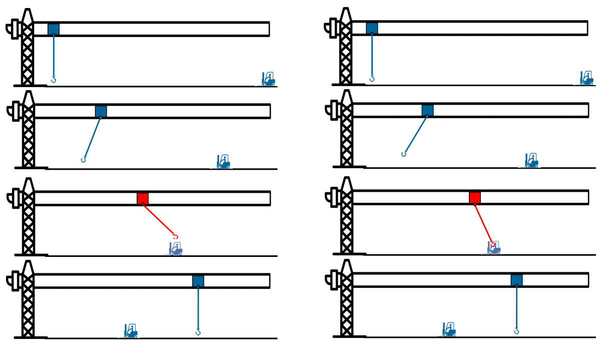


Figure 2: Illustration of the trajectories generated by the open-loop controller for the crane and vehicle example under disturbance-free (left) and perturbed (right) situations MS: Please change color of the third stage in left figure.

Next, we project the unique open loop trajectory on to a nominal trajectory for each individual component. The robustness of the trajectory means that there is a tube around the nominal trajectory. In a second step, we solve a number of local *guaranteed tracking control* problems, where we synthesize correct-by-construction controllers whose objective is to track the nominal trajectory while staying within the tube.

The overall algorithm is more scalable and provides stronger guarantees than each component by itself. Global trajectory planning scales to large state spaces but does not guarantee correctness and each formal controller synthesis problem is solved locally. Empirically, we show that our algorithm is able to generate provably correct feedback controllers for many systems for which neither technique is individually effective. Of course, since we decompose the problem, it is possible that there is no controller for a particular choice of the robustness parameter, or indeed, for other parameters used by the individual tools. In that case, there is an outer loop that searches through the parameter space.

We have implemented our technique based on two ingredients: the ALTRO open-loop trajectory planner [15] and the SCOTS correct-by-construction controller synthesis tool [30]. ALTRO is a state-of-the-art trajectory planning tool based on optimal control. It handles nonlinear dynamics and scales to large dimensions, but ignores disturbances or modeling uncertainties. SCOTS implements a highly parallelizable *abstraction-based controller synthesis* algorithm that generates a provably correct feedback controller for any linear temporal logic specification.

We evaluate our tool empirically on a number of multi-robot benchmarks, including multi-robot co-ordination, formation flight, and lane merging. In each case, we demonstrate that our system can efficiently find decentralized and correct controllers.

Fig. 2 shows a concrete multi-robot reach-avoid scenario with an overhead crane hanging from a trolley along a horizontal rail and a cart that drives underneath the crane in the same horizontal axis. The goal is to move the crane and the vehicle such that they do not collide. Fig. 2 (**left**) shows an animation of a possible open loop behavior from an initial configuration (frame 1) to a final one (frame 4), where the crane and the vehicle have crossed each other. The trajectory is generated by accelerating the trolley, causing the crane to swing up and thus creating enough space for the vehicle to pass. Unfortunately, in the presence of disturbances, such as wind or a slippery floor, a trajectory may not be free of collisions: the same open loop behavior can cause a collision (Fig. 2 **right**). Instead, our algorithm computes a global *robust* trajectory for the system; the robustness parameter ensures a "wider berth." The global trajectory is projected to the crane and the vehicle, and we compute guaranteed tracking controllers that ensure there is no collision despite the disturbances. In our experiments, planning with ALTRO took less than a second and feedback controller synthesis with SCOTS about 10 minutes. At the same time, a global approach to find a correct solution does not scale. The global state space is 10^{10} times larger and SCOTS timed out with 1.5 TB of memory.

Related Work The field of multi-agent planning is too large for a comprehensive survey; we point to very good text books [8, 19, 20, 31] for an introduction. We categorize closely related work into (1) ones combining planning and tracking controller synthesis, (2) ones addressing multi-agent formal controller synthesis, and (3) approaches combining (1) and (2).

Techniques combining high-level planning and low-level tracking are a staple of classical planning and control. More recently, several techniques consider the problem of formal guarantees for such planners. Existing methods can be categorized based on the dynamics that they can handle (e.g., linear or nonlinear), whether the system is subject to disturbances, considered class of specifications, and scalability. A common approach to solve for reach-avoid specifications is to perform the high-level planning over a lower dimensional dynamical model and then use Sum-of-Squares (SOS) programming or Hamilton Jacobi (HJ) or satisfiabiity modulo convex programming (SMC) methods to obtain a low-level controller ensuring a bounded error between the two models [14, 23, 26, 34, 36]. In contrast to [14, 34], our method does not require finding a linear mapping between the low and high dimensional models. **MS:** Nilsson et al. proposed a method that decomposes the state space into a lower-order planning space, and a higher-order internal dynamics space, so that planning can be performed fast and accurate tracking can be achieved using a set of control barrier functions that are computed using SOS method [26]. Despite providing guarantees for the worst-case bounded disturbances, their method is not capable of solving reach-avoid tasks which involve dynamic obstacles as in the multi-agent case. While we choose SCOTS because the algorithm can be effectively parallelized [18], in principle, we could also use SOS or Hamilton Jacobi or SMC approaches. Some other works only consider special classes of systems such as linear ([10, 29, 38]), disturbance-free ([11, 35, 37]), or finite transition systems ([41]). In contrast, our method supports arbitrary nonlinear dynamics and provides a guarantee against worst-case bounded disturbance.

Chen et al. proposed a method, using control barrier functions, that requires some form of inter-robot communication and does not consider external disturbances [7]. Sahin et al. proposed a method that requires the group of robots to be homogeneous [33]. There are methods which do not consider external disturbances and do not provide formal guarantees [16].

Alonso-Mora et al. proposed a method for formation control of a group of communicating homogeneous robots [1]. They first synthesize nominal controller using a fast randomized geometric planning method, namely RRT, and then use optimal control to track the obtained nominal solution. Unlike us, they neither consider external disturbances nor provide formal guarantees. **MS:** Pant et al. considered STL expressable multi-quadrotor missions [27]. They find the reference trajectory by maximizing robustness with respect to the given STL specification, and synthesize tracking controllers so that the reference trajectory is followed closely. Although their method provides some robustness margin, it is not capable of providing guarantee against worst-case bounded additive disturbances as we do. Further, they only consider time-bounded STL specifications. Recently, Xiao et al. [40] proposed a method for synthesis of distributed controllers for a set of autonomous vehicles in a lane merging situation. They use optimal control to compute a nominal controller for the product system, whose goal is to ensure some safety constraints together with ensuring optimality in some respect (like fuel consumption). After that, they use control barrier functions for each robot for tracking the obtained nominal solution. Even though they consider external noise in the system, yet, when there is noise, their controllers can occasionally violate the safety constraints for finite number of time steps. Moreover, they only consider simple linear systems (double integrator) as vehicle models. **MS:** Nikou et al. studied the problem of robust navigation for multi-agent systems [25]. They use optimal control to compute a reference trajectory for the nominal dynamics by neglecting disturbance, and invoke a pre-computed feedback controller to guarantee that the perturbed system's trajectory remains within a bounded hyper-cube around the reference trajectory. In order to avoid collision with obstacles or other agents, their method relies on sensing capabilities of the agents. In contrast, our method does not requires any sensing ability in order to guarantee safe navigation of multi-agents.

There are other works that use a pre-defined motion primitive library to perform planning for multi-robot systems [5, 9, 12, 32]. In contrast, our method directly deals with the dynamical model. Our construction can also be seen as an *assume guarantee* technique that decomposes the global problem based on nominal trajectory tubes; similar decompositions have been studied in the discrete case [2, 21]. The closest related work that match our level of generality is the one by Bansal et al. [4]. However, they assume that each robot has its own reach-avoid specification and they only need to avoid collision with the other robots. In contrast, we allow global reach-avoid specifications, which subsume the specifications that they consider. In fact, there are control problems which cannot be easily expressed in their problem setting, but can be easily expressed in our problem setting: examples are robots maintaining a formation while fulfilling their tasks [1], etc.

2 SYSTEMS AND CONTROLLERS

Notation. We denote the set of natural numbers including zero by $\mathbb{N}$. We use the notation $\mathbb{R}$ and $\mathbb{R}_{>0}$ to denote respectively the set of real numbers and the set of positive real numbers. We use superscript $n > 0$ with $\mathbb{R}$ and $\mathbb{R}_{>0}$ to denote the Cartesian product of n copies of $\mathbb{R}$ and $\mathbb{R}_{>0}$ respectively. Given two points $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{R}^n$ and a relational symbol $\triangleright \in \{ \le, <, =, >, \ge \}$, we write $x \triangleright y$ if $x_i \triangleright y_i$ for every $i \in \{1, 2, \ldots, n\}$. The operator $|\cdot|$ is used to denote both the absolute value of a vector and cardinality of a set, depending on the type of the operand, and the operator $\|\cdot\|$ is used to denote the infinity norm.

Let $f\colon A \to B$ and $g\colon C \to D$ be two functions. We define the product function $f \otimes g\colon A \times C \to B \times D$, $f \otimes g\colon (a, c) \mapsto (f(a), g(c))$. The product is associative and extends to more than two functions in the obvious way. Given $c \in \mathbb{R}^n$ and $\varepsilon \in \mathbb{R}^n_{>0}$, the ball with center c and radius ε in $\mathbb{R}^n$ is denoted by $\Omega_\varepsilon(c) := \{ x \in \mathbb{R}^n \mid |x - c| \le \varepsilon \}$.

Let A be a set. We use the notation A^∞ to denote the set of all finite and infinite sequences formed using the members of A. Our control tasks are defined using a subset of Linear Temporal Logic (LTL). In particular, we use the *until* operator $\mathcal{U}$ and the *next* operator $\bigcirc$ defined as follows. Let p and q be subsets of $\mathbb{R}^n$ and $\rho = (x_0, x_1, \ldots)$ be an infinite sequence of elements from $\mathbb{R}^n$. We write $\rho \models \bigcirc^k p$ if $x_k \in p$. We write $\rho \models p \mathcal{U} q$ if there exists $i \in \mathbb{N}$ s.t. $x_i \in q$ and $x_j \in p$ for all $0 \le j < i$. For detailed syntax and semantics of LTL, we refer to [3] and references therein.

Control Systems. A *control system* $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ consists of a *state space* $X \subseteq \mathbb{R}^n$, an *initial state* $x_{\mathsf{in}} \in X$, an *input space* $U \subseteq \mathbb{R}^m$, a compact *disturbance set* $W \subset \mathbb{R}^n$ containing 0, and a vector field $f : X \times U \to X$ modeling the *nominal dynamics* of the system. A *trajectory* of Σ is a finite or infinite sequence $x_0, x_1, \ldots \in X^\infty$ such that $x_0 = x_{\mathsf{in}}$ and for each $i \ge 0$, there is an $u_i \in U$ and $w_i \in W$ such that $x_{i+1} = f(x_i, u_i) + w_i$. A trajectory is *nominal* if $w_i = 0$ for all $i \ge 0$. Intuitively, a control system represents a (possibly nonlinear) dynamical system with control inputs form U and disturbances from the set W.

Product Control Systems. Let $\{\Sigma^i\}_{i \in [1;N]}$, $\Sigma^i = (X^i, x^i_{\mathsf{in}}, U^i, W^i, f^i)$, for $i \in [1; N]$, be a set of N control systems. The *product control system* of $\{\Sigma^i\}_{i \in [1;N]}$ is defined as the control system $\Sigma^\times = (X^\times, x^\times_{\mathsf{in}}, U^\times, W^\times, f^\times)$, where $X^\times := X^1 \times \ldots \times X^N$, $x^\times_{\mathsf{in}} := (x^1_{\mathsf{in}}, \ldots, x^N_{\mathsf{in}})$, $U^\times := U^1 \times \ldots \times U^N$, $W^\times := W^1 \times \ldots \times W^N$, and $f^\times := f^1 \otimes \ldots \otimes f^N$. We use the state projection operator $\mathrm{proj}^i\colon X^\times \to X^i$ with $\mathrm{proj}^i(x^1, \ldots, x^N) = x^i$. For brevity, we write $\{\Sigma^i\}$ instead of $\{\Sigma^i\}_{i \in [1;N]}$, when the range of i is irrelevant or clear from context.

Sampled-time Systems. We define control systems and their product over discrete time. Such systems can be obtained as time discretizations of continuous-time nonlinear dynamical systems. We sketch the connection. Consider the tuple $(X, x_{\mathsf{in}}, U, W, f)$ as above and suppose that $f : X \times U \to X$ is such that $f(\cdot, u)$ is locally Lipschitz for all $u \in U$. Given a time horizon $\tau > 0$, an initial state x_0, and a constant input u, define the *continuous time trajectory* $\zeta_{x_0,u}$ of the system on the time interval $[0, \tau]$ as an absolutely continuous function $\zeta_{x_0,u} : [0, \tau] \to X$ such that $\zeta_{x_0,u}(0) = x_0$, and $\zeta_{x_0,u}$ satisfies the differential inclusion $\dot{\zeta}_{x_0,u}(t) \in f(\zeta_{x_0,u}(t), u) + W$ for almost all $t \in [0, \tau]$. Given τ, x_0, and u, we define $Sol(x_0, u, \tau)$ as the set of all $x \in X$ such that there is a continuous time trajectory $\zeta_{x_0,u}$ with $\zeta(\tau) = x$. A sequence $x_0, x_1, \ldots$ is a *time-sampled trajectory* if $x_0 = x_{\mathsf{in}}$ and for each $i \ge 0$, we have $x_{i+1} \in Sol(x_i, u_i, \tau)$ for some $u_i \in U$.

Given a continuous-time nonlinear dynamical system with the tuple $(X, x_{\mathsf{in}}, U, W, f)$ as above and a sampling time τ, there exist techniques to construct a control system $\Sigma = (X, x_{\mathsf{in}}, U, W, f_\tau)$ such that every time-sampled trajectory of the continuous-time system is also a trajectory of Σ. We omit the details of the construction; see, e.g., [28].

Controllers. An *open-loop controller* for $\Sigma = (X, x_{\mathsf{in}}, U, W, f)$ over a time interval $[0;T]$ with $T \in \mathbb{N}$ is a function $C\colon [0;T] \to U$. The open-loop is obtained when we connect C with Σ serially, denoted by $C \triangleright \Sigma$. The set of trajectories of the open-loop system $C \triangleright \Sigma$ consists of all finite trajectories $x_0, x_1, \ldots, x_T$ such that $x_0 = x_{\mathsf{in}}$ and for all $i \in [0;T-1]$, we have $x_{i+1} = f(x_i, C(i)) + w_i$ for some $w_i \in W$.

A *feedback controller* for Σ over a time interval $[0;T]$, $T \in \mathbb{N}$, is a function $C\colon X \times [0;T] \to U$. We denote the feedback composition of Σ with C as $C \parallel \Sigma$. The set of trajectories of the closed-loop system $C \parallel \Sigma$ consists of all finite trajectories $x_0, x_1, \ldots, x_T$ such that $x_0 = x_{\mathsf{in}}$ and for all $i \in [0;T-1]$, we have $x_{i+1} = f(x_i, C(x_i, i)) + w_i$ for some $w_i \in W$. For both open-loop and feedback composition, the nominal trajectories are those trajectories for which $w_i = 0$ for all $i \geq 0$.

Now let $\{\Sigma^i\}$ be a set of control systems. We can define *global* open-loop and feedback controllers by defining the respective controllers on the product system $\Sigma^\times$. We can also define *local* open-loop and feedback controllers C^i for each Σ^i. In this latter case, the set of trajectories of the system $\{C^i\} \triangleright \{\Sigma^i\}$ (respectively, $\{C^i\} \parallel \{\Sigma^i\}$) are finite sequences $x_0^\times, x_1^\times, \ldots, x_T^\times$ such that $x_0^\times = x_{\mathsf{in}}^\times$ and for each $j \in [0;T-1]$, we have $\mathrm{proj}^i(x_{j+1}^\times) = f^i(\mathrm{proj}^i(x_j^\times), C^i(j)) + w_{ji}$ (respectively, $\mathrm{proj}^i(x_{j+1}^\times) = f^i(\mathrm{proj}^i(x_j^\times), C^i(\mathrm{proj}^i(x_j^\times), j)) + w_{ji}$) for some $w_{ji} \in W^i$, for each $i \in [1;N]$.

Decentralized Controller Synthesis Problem. Let $\{\Sigma^i\}$ be a set of control systems. A (global) *control specification* $\mathcal{L}$ is a set of finite sequences in $(X^\times)^*$. Intuitively, a control specification specifies a set of "good behaviors" of the product system. The *decentralized open-loop (resp. feedback) controller synthesis problem* asks, given $\{\Sigma^i\}$ and a global control specification $\mathcal{L}$, to construct a set of local open-loop (resp. feedback) controllers $\{C^i\}$ such that the trajectories of $\{C^i\} \triangleright \{\Sigma^i\}$ (resp. $\{C^i\} \parallel \{\Sigma^i\}$) all belong to the global control specification $\mathcal{L}$.

In particular, we consider *reach-avoid* specifications, written as $\neg \mathit{Avoid}\, \mathcal{U}\, \mathit{Goal}$ in linear temporal logic, for two subsets $\mathit{Avoid}, \mathit{Goal} \subseteq X^\times$. A trajectory $x_0, x_1, \ldots$ satisfies the reach avoid specification if there is some $j \in \mathbb{N}$ such that $x_j \in \mathit{Goal}$ and for all $i \in [0; j-1]$, we have $x_i \notin \mathit{Avoid}$.

Our notion of product of control systems is a Cartesian product of each component. Thus, we assume that the dynamics of individual control systems are not coupled. However, the overall trajectories of the control systems can be coupled through the control objective. Indeed, our choice of the specification Φ on the product state space $X^\times$ subsumes many interesting class of control tasks. For example, we can express situations when the robots have their own local reach-avoid specifications, and they need to avoid collision among each other; we consider such a specification in our examples in Sec. 4.1. In addition, we can also specify more general tasks which cannot be easily decomposed into individual subtasks for the robots. One example is the *formation control problem*, which we consider in the example in Sec. 4.2, where a set of robots need to reach some location while maintaining a given geometric formation. Since the formation is defined using the relative positions of the robots, it is not possible to decompose this task into separate local reach-avoid subtasks.

Example 2.1. Consider our motivating example in which there are two control systems: the crane and the factory vehicle. The detailed models for both of the systems are given in Sec. 4.1.2. The crane starts at $x_{\mathsf{in}}^1 = \begin{bmatrix}0 & 0 & \pi & 0\end{bmatrix}^T$, while the vehicle starts at $x_{\mathsf{in}}^2 = \begin{bmatrix}8 & 0\end{bmatrix}^T$. The goal set for the crane is $\mathit{Goal}^1 = \Omega_{r_1}(g_1)$, which is a ball centered at $g_1 = \begin{bmatrix}5 & 0 & \pi & 0\end{bmatrix}^T$ with a desired size $r_1 \in \mathbb{R}^4_{>0}$. The goal set for the vehicle is similarly defined as $\mathit{Goal}^2 = \Omega_{r_2}(g_2)$ with center $g_2 = \begin{bmatrix}4 & 0\end{bmatrix}^T$ and size $r_2 \in \mathbb{R}_{>0}$. The goal set over the product space is defined as $\mathit{Goal} = \mathit{Goal}^1 \times \mathit{Goal}^2 \subseteq \mathbb{R}^6$. The set Avoid is defined as $\mathit{Avoid} = \{x^\times \in X^\times \mid D(x^1, x^2) \leq \delta\}$ where x^1 and x^2 denote components of $x^\times$ corresponding to the two systems, D represents geometric distance between the two systems in the two-dimensional space and δ denotes a minimum safety distance requirement. It can be noticed that despite the fact that dynamics of the two systems are not coupled, to fulfill the reach-avoid specification $\neg \mathit{Avoid}\, \mathcal{U}\, \mathit{Goal}$, one has to take both of the dynamics into account. □

3 MULTI-AGENT REACH-AVOID SYNTHESIS

3.1 Problem Statement

We now consider the decentralized controller synthesis problem for a set of control systems $\{\Sigma^i\}$ w.r.t. a global reach-avoid specification $\Phi = \neg \mathit{Avoid}\, \mathcal{U}\, \mathit{Goal}$, where $\mathit{Avoid}, \mathit{Goal} \subseteq X^\times$ are subsets of the product state space.

First, we define a *robust version* of the control specification. Let $\varepsilon \in \mathbb{R}^n_{>0}$ be a *robustness margin.* We define the ε-robust version of Φ, denoted by $\Phi_\varepsilon := (\neg \mathit{Avoid}'\, \mathcal{U}\, \mathit{Goal}')$, where $\mathit{Avoid}' = \mathit{Avoid} \oplus \Omega_\varepsilon(0)$ and $\mathit{Goal}' = \mathit{Goal} \ominus \Omega_\varepsilon(0)$, and $\oplus$ and $\ominus$ are set operators denoting the Minkowski addition and difference, respectively. Intuitively, if a trajectory $x_0, x_1, \ldots$ satisfies Φ_ε, then any trajectory $y_0, y_1, \ldots$ such that $|x_i - y_i| \leq \varepsilon$ satisfies Φ.

Problem 1 (Decentralized Controller Synthesis). ***Inputs:*** *Control systems $\Sigma^i = (X^i, x_{\mathsf{in}}^i, U^i, W^i, f^i)$, $i \in [1;N]$, and global specification $\Phi = \neg \mathit{Avoid}\, \mathcal{U}\, \mathit{Goal}$.*
Parameters: *A robustness margin $\varepsilon \in \mathbb{R}^n_{>0}$.*
Outputs: *Local feedback controllers $\{C^i\}$ for $\{\Sigma^i\}$, $i \in [1;N]$, such that $\{C^i\} \parallel \{\Sigma^i\}$ realizes Φ.*

It is important to notice that any solution for this problem is required to provide a *formal guarantee* on the satisfaction of Φ, i.e., the reach-avoid specification Φ *must* be satisfied under any disturbance affecting the control systems. **MS:** Further, the solution must not require any information exchange between the different agents. Embedding this feature simplifies implementation by eliminating the need for regular synchronization between agents during runtime.

To solve the decentralized control problem, we propose to compute a nominal trajectory over the simplified dynamical model of the product system and then to employ a guaranteed method to synthesize controllers locally that account for model mismatches. By simplified, we mean absence of disturbance, i.e., we set $W = \{0\}$.

Algorithm 1 Multi-agent Controller Synthesis

(1) For every i, let $\Sigma^i_{nom} = (X^i, x^i_{\text{in}}, U^i, \{0\}, f^i)$ be the nominal control system of Σ^i that ignores the disturbance. Compute the product control system $\Sigma^{\times}_{nom}$ of $\{\Sigma^i_{nom}\}$. Use a scalable *planner* to compute a nominal open-loop controller $C^{\times}_{nom} : [0;T] \to U^{\times}$ such that the specification Φ_{ε} is satisfied by $C^{\times}_{nom} \triangleright \Sigma^{\times}_{nom}$. Note that T is the first time the set $Goal'$ is visited.

(2) Decompose $C^{\times}_{nom}$ into *local* open-loop controllers $\{C^i_{nom}\}$ for the set of $\{\Sigma^i_{nom}\}$ by projecting the output of $C^{\times}_{nom}$ into local input spaces U^i. Further, for every i, find the unique nominal open-loop trajectory $\rho^i = (x^i_{0,nom}, \dots, x^i_{T,nom})$ of $C^i_{nom} \triangleright \Sigma^i_{nom}$. These trajectories are unique since there is no disturbance.

(3) Let $\varepsilon^i \in \mathbb{R}^{n_i}_{>0}$, $i \in [1;N]$, be the projections of ε compatible with the state dimensions of Σ^i. Each control system Σ^i uses a *guaranteed tracking* method to compute a closed-loop controller C^i such that $C^i \parallel \Sigma^i$ *tracks* the nominal trajectory ρ^i and stays within its ε^i-neighborhood, i.e., $C^i \parallel \Sigma^i$ satisfies the specification

$$\Phi^i_{track} := \bigwedge_{k\in[0;T]} \bigcirc^k \Omega_{\varepsilon^i}(x^i_{k,nom}). \quad (1)$$

We summarize our approach for solving Prob. 1 in Alg. 1. The approach is composed of three main steps: (1) Synthesize a global open-loop controller for the *nominal system* as a single planner task on the product system to satisfy Φ_ε; (2) Project the controller into local controllers and obtain a nominal trajectory for each system; and (3) Design local closed-loop controllers to track the nominal trajectory while always staying within the robustness margin. The soundness of the technique is summarized below.

Theorem 3.1. *Local feedback controllers $\{C^i\}$ synthesized by Alg. 1 guarantee that the global specification is satisfied by the product system $\{C^i\} \parallel \{\Sigma^i\}$.*

Proof. Note that Φ_ε is a stronger version of Φ and is intentionally made conservative to allow for ε-deviation in the trajectory of the product system. Since $\{\rho^i\}$ is the unique solution of the nominal product system and satisfies Φ_ε, it is guaranteed that ε-perturbation of this nominal trajectory satisfies Φ. It can be observed that all solutions of Σ^i stay within distance ε^i of the nominal trajectory ρ^i regardless of the disturbance. This completes the proof. □

Next, we discuss our implementation of Alg. 1 for the global *open-loop planner* (Step 1) and the local *guaranteed trajectory tracking* (Step 3). Note that Step 2 is a simple projection from the product space into local spaces. While we instantiate particular techniques, our method can be used with other implementations as well.

3.2 Open-loop Planning

The planner used for generating nominal trajectories in Step 1 of our algorithm should be fast and scalable. In addition, it should be capable of handling non-linear dynamics and constraints. Our choice for the planner is ALTRO [15]. ALTRO is a fast and numerically robust solver for constrained trajectory optimization problems and and is capable of handling nonlinear state and input constraints. Given a product system $\Sigma^{\times}_{nom}$, reach-avoid specification Φ_ε and time horizon T, ALTRO computes an open-loop controller $C^{\times}_{nom} : [0;T] \to U^{\times}$ by solving the optimization

$$\begin{aligned} \underset{u_0^\times, u_1^\times, \dots, u_T^\times}{\text{minimize}} \quad & \ell_T(x_T^\times) + \sum_{k=0}^{T-1} \ell_k(x_k^\times, u_k^\times) \\ \text{subject to} \quad & x_{k+1}^\times = f^\times(x_k^\times, u_k^\times), \quad \forall k \in [0;T-1] \\ & g(x_k^\times, u_k^\times) \le 0, \qquad \forall k \in [0;T] \\ & h(x_k^\times, u_k^\times) = 0, \qquad \forall k \in [0;T], \end{aligned}$$

where $\ell_k(\cdot,\cdot)$ denotes a quadratic objective function assigning cost to each pair of state and input before the end of horizon, $\ell_T(\cdot)$ represents a quadratic objective function assigning penalty to the final state $x_T^\times$ being away from the goal set $Goal'$. The constraints $g(x_k^\times, u_k^\times) \le 0$ and $h(x_k^\times, u_k^\times) = 0$ capture the requirement that at each time k the state should not be in $Avoid'$, the state $x_T^\times$ should be in $Goal'$, and the input $u_k^\times$ should always be in $U^\times$. In multi-robot scenarios, the inequality constraints can be used to define collision and obstacle avoidance specifications and the equality constraints can define fixed formation specification. **MS:** Note that the reach-avoid specification is fulfilled if the corresponding equality and inequality constraints (i.e., $g(\cdot) \le 0$, $h(\cdot) = 0$) are satisfied at every time-step and thus choice of the quadratic objective function (ℓ_k for $k \in [0;T]$) is not crucial.

Remark 1. *ALTRO supports only bounded horizon control problem. For this reason, we model the states in $Goal'$ as a sink state and select a time horizon T for solving the planning task on the nominal product system. We increase the horizon T if ALTRO is not able to find a controller. We stress that this is an ALTRO-specific implementation detail, and our overall method does not rely on a fixed time horizon.*

3.3 Guaranteed Trajectory Tracking

Trajectories computed in the planning stage might not be followed in the presence of disturbance and therefore we need to use a formally guaranteed tracker to satisfy the given reach-avoid specification. Our choice for this purpose is the so-called abstraction-based controller design (ABCD). ABCD can handle nonlinear dynamics, (bounded) uncertainties and ω-regular specifications. In particular, we use the SCOTS tool [30] for implementing ABCD. Next, we introduce briefly the basics of ABCD. For simpler notation, we omit the control system index i in the rest of this section.

Finite-state Abstraction of Control Systems. Let $\Sigma = (X, x_{\text{in}}, U, W, f)$ be a control system and $Domain$ be a subset of X which imposes a safety specification on all possible trajectories of the system. Let $\widehat{X}$ be a given *finite partition* of $Domain$, and $\widehat{U}$ be a *finite subset* of equally spaced (w.r.t. infinity norm on $\mathbb{R}^m$) points in the input set U. A *finite-state abstraction* of Σ is a finite state-transition system $(\widehat{X}, \widehat{U}, \widehat{f})$, where $\widehat{x}'$ is in $\widehat{f}(\widehat{x}, u)$ if there is a pair of states $x \in \widehat{x}$ and $x' \in \widehat{x}'$ such that $x' \in Sol_\Sigma(x, u)$.

In this work, we will use uniformly sized rectangular partition elements to construct the set $\widehat{X}$ from the set $Domain$. Without going into the detail of the construction, we assume that the size of the partition elements is provided as a vector $\eta_x \in \mathbb{R}^n_{>0}$ which is an input to the abstraction procedure. Note that, the larger η_x is (where comparison is made dimension-wise), the smaller is the state space $\widehat{X}$ resulting in an efficient computation. On the other hand, the smaller η_x is, the better is the precision of the abstraction $\widehat{\Sigma}$ increasing the chance of a successful controller synthesis. Similar to the state space partition size η_x, we also assume that the set $\widehat{U}$ is chosen based on an input space discretization parameter $\eta_u \in \mathbb{R}^m_{>0}$ that governs the distance between the points in $\widehat{U}$.

Feedback Refinement Relation. Let Σ be a control system and $\widehat{\Sigma}$ be its finite-state abstraction. A *feedback refinement relation* (FRR) from Σ to $\widehat{\Sigma}$ is a relation $Q \subseteq Domain \times \widehat{X}$ s.t. for all $x \in Domain$ there is some $\widehat{x} \in \widehat{X}$ such that $Q(x, \widehat{x})$ and for all $(x, \widehat{x}) \in Q$, we have (i) $\widehat{U}_{\widehat{\Sigma}}(\widehat{x}) \subseteq U_{\Sigma}(x)$, and (ii) $u \in U_{\widehat{\Sigma}}(\widehat{x}) \Rightarrow Q(f(x,u)) \subseteq \widehat{f}(\widehat{x}, u)$, where $U_{\Sigma}(x) := \{ u \in U \mid f(x,u) \neq \emptyset \}$ and $\widehat{U}_{\widehat{\Sigma}}(\widehat{x}) := \{ u \in \widehat{U} \mid \widehat{f}(\widehat{x}, u) \neq \emptyset \}$. We write $\Sigma \preceq_Q \widehat{\Sigma}$ if Q is an FRR from Σ to $\widehat{\Sigma}$.

Abstraction-based Controller Design. The abstraction-based controller design (ABCD) [28] is a 3-step method to find a robust controller for the control system Σ: First, we compute a finite state abstraction $\widehat{\Sigma}$ s.t. $\Sigma \preceq_Q \widehat{\Sigma}$. Second, we synthesize an abstract controller of the form $\widehat{C} : \widehat{X} \to \widehat{U}$ for $\widehat{\Sigma}$ using methods from the reactive synthesis literature. Finally, we obtain the desired controller C as $C := \widehat{C} \circ Q$. It is known that this three step process produces a feedback controller C such that $C \parallel \Sigma$ satisfies the specification [28].

If a controller cannot be found, we reduce the discretization parameters η_x and η_u and try again, or use a larger robustness margin ε.

Optmization: Local ABCD around the nominal trajectory The abstraction process of ABCD usually requires computation of abstract transitions over the whole compact set *Domain*, which is computationally expensive. Luckily, for Step 3 of Alg. 1, we only need to compute transitions in the ε-neighborhood of the given nominal trajectory. Given a control system Σ, together with a reference open-loop trajectory $\rho = (x_{0,nom}, \ldots, x_{T,nom})$ and a tube size $\varepsilon \in \mathbb{R}^n_{>0}$, we iteratively construct a tube as union of ε-balls around the reference trajectory (note that we have omitted the system index i for simpler notation). Next, we compute finite state abstraction for Σ for the chosen parameters η_x, η_u by setting

$$Domain := \bigcup_{k=0}^{T} \Omega_{\varepsilon}(x_{k,nom}).$$

Experimentally, this local domain is key to scalability.

Remark 2. *Our decentralized controller synthesis approach features an interplay between the global open-loop planning and local formal synthesis via the robustness parameter ε and the discretization parameters η_x, η_u. The parameter ε should be large enough to allow deviation from the nominal trajectory caused by the disturbance. A small ε makes the local specification Φ^i_{track} very strong for the ABCD synthesis thus requires large computational complexity with smaller discretization parameters η_x, η_u. On the other hand, large ε makes the specification Φ_ε very conservative or infeasible for the global open-loop planning. Therefore, appropriate parameters should be selected iteratively for successful application of our synthesis approach.*

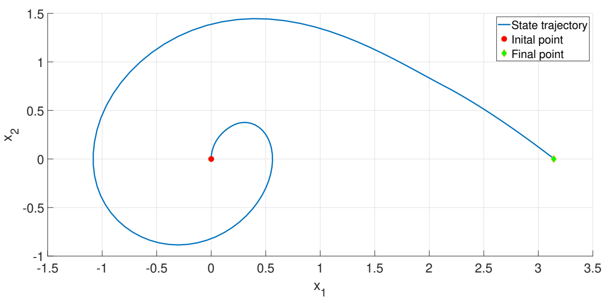


Figure 3: The nominal trajectory for the inverted pendulum.

3.4 Hybrid vs Geometric Planning

Note that Step 3 of Alg. 1 does not use the nominal controller obtained in Step 1 and requires only the nominal trajectories. Then one could argue that, instead of using ALTRO to generate nominal trajectories, a fast *geometric planner* [17] can be employed to generate geometric plans. However, fast geometric planners do not usually take into consideration the dynamics and control constraints. In our experience, the plans for nominal trajectories generated while ignoring the system dynamics are often untrackable unless the underlying system has special properties (e.g., differential flatness [24]). This is especially true for systems with restricted control capabilities or under-actuated systems. We demonstrate this phenomenon on a control system Σ that is a simple 2-dimensional pendulum with the following nominal dynamics:

$$\dot{x}_1 = x_2 \quad \dot{x}_2 = -\sin(x_1) + u/5,$$

where x_1 represents the angle (in Radian) of the pendulum rod measured counter-clockwise from the vertical upright position, and x_2 represents the rate of change of x_1 or the angular velocity. Suppose the initial state of the pendulum is $(\pi, 0)$, i.e., when the pendulum is in the vertical downward position and is stationary. Suppose we want to find a controller for the goal $Goal = \{ (0,0) \}$, i.e., when the pendulum is in the vertical upward position and is stationary. The set of unsafe states *Avoid* is empty, i.e., no safety constraint is imposed. When we use ALTRO to compute an open-loop controller C, unsurprisingly, the controlled trajectory of $C \triangleright \Sigma$ looks like a spiral, as shown in Fig. 3. The synthesis of tracking controller using ABCD is indeed successful when we feed this nominal trajectory to our ABCD solver. However, if we use a geometric planner for this example that ignores the dynamics, the nominal trajectory would be a straight line path from $(\pi, 0)$ to $(0, 0)$, which gives an infeasible tracking problem for ABCD, due to the restrictions on possible trajectories of the pendulum coming from the its dynamics.

4 EXPERIMENTAL EVALUATION

We evaluate an implementation of Alg. 1 on two distinct categories of problems: local reach-avoid problems with collision avoidance

Table 1: Runtimes for four case studies. Run times (in seconds) for computing open-loop controllers over the corresponding product spaces using ALTRO (T_{AL}), number of input-state pairs of the finite abstraction for the largest ABCD task (N_l), abstraction and synthesis times in SCOTS for that task (respectively T_l^{abs} and T_l^{syn}), number of input-state pairs of the finite state abstraction for the global ABCD (N_g), abstraction and synthesis times for computing a global controller for the product system using SCOTS (respectively T_g^{abs} and T_g^{syn}).

Case-study	Global planning	Local ABCD			Global ABCD		
	T_{AL}	N_l	T_l^{abs}	T_l^{syn}	N_g	T_g^{abs}	T_g^{syn}
Multi-drone path planning	125	1.15×10^8	47.66	7.9	2.73×10^{104}	OOM	OOM
Crane and vehicle	0.65	8.56×10^8	511	91	2.16×10^{18}	OOM	OOM
Lane merging	100	1.01×10^8	37.60	9.51	3.84×10^{65}	OOM	OOM
Multi-drone formation control	163	4.02×10^8	69.78	13.6	3.7×10^{59}	OOM	OOM

and global formation control problems. We consider four case studies: multi-drone path planning, crane and vehicle, lane merging, and multi-drone formation control. The design of nominal controller was performed on a laptop with core i7-4510u CPU at 3.10GHz, with 8GB of RAM. The formal controller synthesis was performed on a cluster with 4 Intel Xeon E7-8857 v2 CPUs (48 cores in total) at 3GHz, with 1.5TB of RAM.

In all of our case studies, robots are moving in a two-dimensional space that has obstacles. Table 1 shows run times for different stages of each experiment. For local ABCD, the reported numbers correspond to the maximum value among all of the agents. This choice is due to the fact that feedback controllers for different agents can be computed independently in parallel over different machines. To provide a more fine-grained comparison, Fig. 4 shows the variations of run times of the different local ABCD tasks for every experiment. We have excluded the crane and vehicle case study in the figure due to an expected large variance originating from different dynamics. **MS:** The corresponding numbers will be reported in the next section. Notice that a higher number of input-state pairs does not necessarily result in a higher run time for local ABCD as the number of transitions and features of the parallel implementation can play a role. We compare our approach with the ABCD applied to the product system to satisfy the global specification. As reflected in Table 1, memory requirement for the global ABCD exceeds our system's limit (1.5TB of RAM) in all of the experiments.

4.1 Local reach-avoid with collision avoidance

We first consider situations when each robots has an individual reachability specification, and they need to ensure a minimum safe distance from each other and the obstacles.

Suppose $\{\Sigma^i\}$ models a set of robots, $Goal^i \subseteq X^i$ are the individual goal sets, and $\delta \in \mathbb{R}_{>0}$ is a safety margin. Suppose the specification requires that each robot Σ^i eventually reaches $Goal^i$ while avoiding the obstacle $Obs \subseteq \mathbb{R}^2$ and collision with robots by the margin δ. The global specification on the product system $\Sigma^\times$ is as follows:

- The goal set $Goal \subseteq X^\times$ is defined as $Goal := Goal^1 \times \ldots \times Goal^N \subseteq X^\times$, and

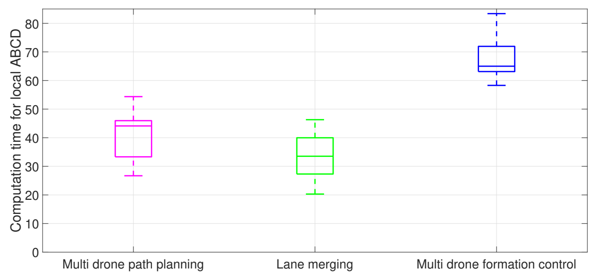


Figure 4: Variations of run times of local ABCD among different agents for three case studies. MS: add time unit to the vertical axis.

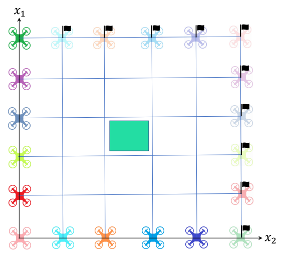
- The avoid set $Avoid \subseteq X^\times$ is defined as

$$\left\{ x^\times \in X^\times \;\middle|\; \begin{array}{c} \exists i \in [1;N]\,.\, D(x^i, Obs) \le \delta \\ \vee \\ \exists i,j \in [1;N]\,.\, i \neq j\,.\, d\left(x^i, x^j\right) \le \delta \end{array} \right\}, \quad (2)$$

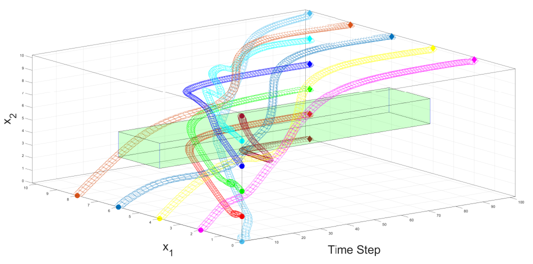
where x^i denotes the component of $x^\times$ corresponding to Σ^i, $d(\cdot,\cdot)$ denotes a predicate for measuring the geometric distance between positions of two systems located in the two-dimensional space, $D(\cdot,\cdot)$ denotes a predicate for measuring the geometric distance between position of one system and the obstacle $Obs \subseteq \mathbb{R}^2$.

In this category of problems, we apply our approach to three case studies as discussed next.

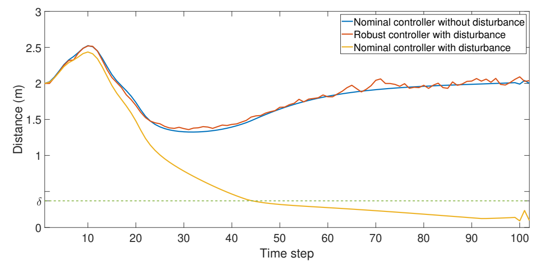
4.1.1 Multi-drone path planning. We consider a planning scenario for ten identical drones ($N = 10$). A set of initial and target states for each drone are depicted in Fig. 5a. The drones must avoid hitting a physical obstacle that is the green square in Fig. 5a. The control objective is to synthesize a feedback controller for each drone so that in the presence of (bounded) disturbance, beginning from the specified initial state, the corresponding target state is reached within a finite horizon, while avoiding collision with other drones and the physical obstacle at every time point.



(a)



(b)



(c)

Figure 5: (a) The mission map for the multi-drone path planning example (b) Time-state space illustration of tubes enclosing nominal trajectories for the multi-drone path planning (c) Performance of open-loop and feedback controllers in regulating distance between drones for disturbance-free and perturbed situations for the multi-drone path planning example

Every drone is modeled as tuple $\Sigma^i = (X, x^i_{\text{in}}, U, W, f^d_\tau)$. The continuous time dynamics for each drone takes the form

$$f^d(x(t), u(t)) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} u_1 cos(x_3) \\ u_1 sin(x_3) \\ u_2 \end{bmatrix}.$$

where x_1 and x_2 denote the drone's position in two-dimensional space, x_3 denotes the rotational angle, and u_1 and u_2 represent control inputs for each drone. Choosing a sampling time $\tau = 0.1s$, the nominal dynamics f^d_τ can be characterized uniquely. The disturbance set and robustness margin are chosen as $|W| \leq \begin{bmatrix} 0 & 0.025 & 0.025 \end{bmatrix}^T$ and $\varepsilon = \begin{bmatrix} 0.16 & 0.16 & 0.18 \end{bmatrix}^T$.

Selecting the horizon length as $T = 102$ and minimum safe distance as $\delta = 0.28$ (chosen to be compatible with ε), ALTRO computes a valid open-loop trajectory in 125 seconds for the product system with 30 state and 20 input variables. Fig. 5b gives time-space illustration for the safe tubes around the nominal trajectories. We consider state and input spaces to be $X = [-1, 11]^2 \times [-2, 3.3]$ and $U = [-2.4, 2.4]^2$, respectively. Choosing $\eta_X = \begin{bmatrix} 0.02 & 0.02 & 0.02 \end{bmatrix}^T$ and $\eta_U = \begin{bmatrix} 0.3 & 0.3 \end{bmatrix}^T$, Tab. 1 illustrates list of run times and number of input-state pairs corresponding to local and global ABCD. Noticeably, for number of agents $N > 1$, memory requirement for global ABCD exceeds our limits (1.5TB of RAM).

On the other hand, using ALTRO alone would not provide guarantee against bounded disturbance. Fig. 5c illustrates performance of open-loop and feedback controllers on regulating distance between the two of the drones with and without disturbance. As expected, in the absence of disturbance, the open-loop controller suffices and distance between the two drones (shown in solid blue) does not go below the defined threshold. Next, we consider the case when constant additive disturbance vectors $\begin{bmatrix} 0 & 0.025 & 0.025 \end{bmatrix}^T$ and $\begin{bmatrix} 0 & -0.025 & -0.025 \end{bmatrix}^T$ are being applied to the two drones throughout the whole horizon. It can be noticed that applying the open-loop controller causes that distance between the two drones (shown in solid yellow) to go below the predefined threshold. However, the feedback controller is capable of maintaining distance (shown in solid red) within the safe region when the same disturbance is being applied.

4.1.2 *Crane and Vehicle.*

The goal of this example is to study performance of our method for controlling a number of robots with different dynamics. We model the crane and vehicle by tuples $\Sigma^1 = (X^1, x^1_{\text{in}}, U^1, W^1, f^c_\tau)$ and $\Sigma^2 = (X^2, x^2_{\text{in}}, U^2, W^2, f^l_\tau)$, respectively. The crane is modeled as cart-pole system [6]:

$$\ddot{\theta} = \frac{M_t g \sin(\theta) - \cos(\theta)(F + M_p l \dot{\theta}^2 \sin(\theta))}{l(4/3M_t - M_p \cos^2(\theta))} = f^c_1(\theta, \dot{\theta}, F)$$
$$\ddot{z} = \frac{F + M_p l \dot{\theta}^2 \sin(\theta) - M_p l \ddot{\theta} \cos(\theta)}{M_t} = f^c_2(\theta, \dot{\theta}, F),$$

where $g = -9.8\ m/s^2$ is the acceleration of gravity, $M_c = 1$ kg is the cart mass, $M_p = 0.1$ kg is the pole mass, $M_t = M_c + M_p$ denotes the total mass, and $l = 0.5$ m is the half-pole length. Further, the cart's position, the pole's angle, and input force to the cart are denoted by $x^{(1)}_1 = z$, $x^{(1)}_3 = \theta$, and $u^{(1)} = F$, respectively. The continuous-time dynamics of the crane is of the following form:

$$f^c(x^{(1)}(t), u^{(1)}(t)) = \begin{bmatrix} \dot{x}^{(1)}_1 \\ \dot{x}^{(1)}_2 \\ \dot{x}^{(1)}_3 \\ \dot{x}^{(1)}_4 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} x^{(1)}_2 \\ f^c_1(x^{(1)}_3, x^{(1)}_4, u^{(1)}) \\ x^{(1)}_4 \\ f^c_2(x^{(1)}_3, x^{(1)}_4, u^{(1)}) \end{bmatrix}.$$

The vehicle's continuous-time dynamics takes the form of

$$f^l(x^{(2)}(t), u^{(2)}(t)) = \begin{bmatrix} \dot{x}^{(2)}_1 \\ \dot{x}^{(2)}_2 \end{bmatrix} = \begin{bmatrix} x^{(2)}_2 \\ u^{(2)} \end{bmatrix},$$

where $x^{(2)}_1$ and $x^{(2)}_2$ denote the vehicle's position and speed and $u^{(2)}$ represents the vehicle's control input (acceleration). On fixing the sampling time $\tau = 0.1s$, one can derive f^c_τ and f^l_τ. For the crane, the disturbance set and robustness margin are chosen as $|W^1| \leq \begin{bmatrix} 0 & 0.05 & 0 & 0 \end{bmatrix}^T$ and $\varepsilon^1 = \begin{bmatrix} 0.135 & 0.385 & 0.176 & 0.768 \end{bmatrix}^T$. Similarly, for the vehicle, disturbance set and robustness margin are chosen as $|W^2| \leq \begin{bmatrix} 0 & 0.1 \end{bmatrix}^T$ and $\varepsilon^2 = \begin{bmatrix} 0.08 & 0.12 \end{bmatrix}$.

There is no obstacle for this example and for minimum distance between the crane and the vehicle we choose $\delta = 0.035$ (chosen to be compatible with ε^1 and ε^2). Fixing the horizon length to $T = 70$, ALTRO was capable of generating a valid nominal trajectory in 0.65 seconds. Fig. 2 (**left**) demonstrates snapshots of the produced trajectory. Surprisingly enough, applying (constant) additive disturbance

$W = \begin{bmatrix} 0 & 0.05 & 0 & 0 \end{bmatrix}^T$ (to the cart-pole system) causes collision between the crane and vehicle before the end of the mission (Fig. 2 (**right**)).

In the next step, we use SCOTS in order to compute a feedback controller tolerating disturbances W^1 and W^2. We choose state and input spaces for the crane to be $X^1 = [-0.195, 5.49] \times [-1.99, 4.37] \times [1.20, 4.68] \times [-5.44, 5.28]$ and $U^1 = [-7, 7]$, respectively. For the vehicle, we set $X^2 = [3, 9] \times [-3, 1.995]$ and $U^2 = [-3, 3]$. We choose state and input partition sizes $\eta_X^1 = \begin{bmatrix} 0.015 & 0.035 & 0.016 & 0.064 \end{bmatrix}^T$, $\eta_U^1 = 0.2$, $\eta_X^2 = \begin{bmatrix} 0.01 & 0.015 \end{bmatrix}^T$ and $\eta_U^2 = 0.1$. Tab. 1 illustrates list of run times and number of input-state pairs corresponding to local and global ABCD. Noticeably, for the cart-pole model, memory requirement for global ABCD exceeds our limits (1.5TB of RAM). **MS:** Note that computing feedback controllers for the crane and vehicle takes ?? and ?? seconds, respectively. The large difference is due to the difference in the size of transition systems for the two agents. Note that, using ALTRO alone would not provide guarantee against bounded disturbance (Fig 2, **right**).

4.1.3 Lane Merging. Finally, we study a lane merging problem wherein multiple controlled vehicles are driving over two merging lanes (Fig. 6, **top** frame). A dangerous situation may occur at the merging point of the two lanes if vehicles are not controlled properly. Different variants of this problem has been studied in the literature (see, e.g., [39, 40]). Without seeking to optimize fuel consumption or travel time, we set the goal to control the vehicles to pass the merging zone safely. In particular, consider a situation where initially three cars are driving on each of the two lanes (Fig. 6, (**top**)). The control objective for each vehicle is to pass the red dashed line within a finite horizon without hitting the road's sides or colliding with other vehicles.

The nominal dynamics for each of the vehicles is the same as the one for modeling drones (given in Sec. 4.1.1). The disturbance set and robustness margin are chosen as $|W| \le \begin{bmatrix} 0.03 & 0.03 & 0.03 \end{bmatrix}^T$ and $\varepsilon = \begin{bmatrix} 0.16 & 0.16 & 0.16 \end{bmatrix}^T$. For collision and obstacle avoidance, we choose $\delta = 0.37$ (in compliance with ε). The horizon length is fixed to $T = 110$. Given these settings, ALTRO generates a valid nominal trajectory in 100 seconds. Next, we use SCOTS in order to compute feedback controllers tolerating additive disturbance W. We choose state and input spaces for each vehicle's model to be $X = [-0.5, 15] \times [0.1, 7.4] \times [-1, 0.4]$ and $U = [-0.9, 3] \times [-2.1, 2.1]$, respectively. State and input partition sizes are chosen as $\eta_X = \begin{bmatrix} 0.02 & 0.02 & 0.02 \end{bmatrix}^T$ and $\eta_U = \begin{bmatrix} 0.3 & 0.15 \end{bmatrix}^T$. Tab. 1 illustrates list of run times and number of input-state pairs corresponding to local and global ABCD. Noticeably, for number of agents $N > 1$, memory requirement for global ABCD exceeds our limits (1.5TB of RAM). Fig. 6 demonstrates snapshots of one sample trajectory when feedback controllers are employed under the presence of disturbance. It should be noticed that using ALTRO alone would not provide guarantee against bounded disturbance. Fig. 5c illustrates the fact that open-loop controller fails in keeping one of the vehicles away from the road's sides under perturbed situation when constant additive disturbance vector $\begin{bmatrix} -0.03 & 0.03 & -0.03 \end{bmatrix}^T$ is being applied throughout the whole horizon. In contrast, emplying feedback controller results in successful lane merging.

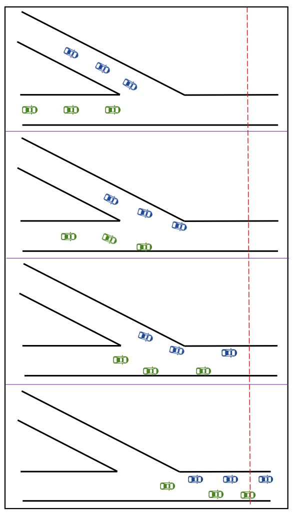


Figure 6: Illustration of a sample trajectory generated by formally guaranteed controllers for the lane merging example

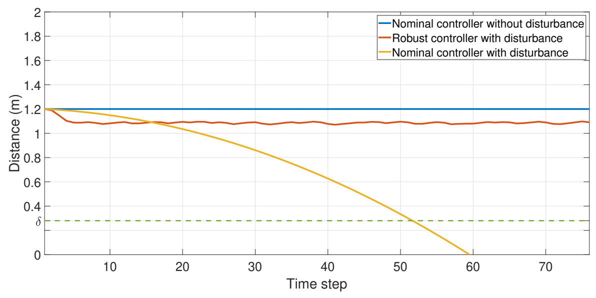


Figure 7: Comparison of open-loop and feedback controllers for the lane merging example

4.2 Global formation control problem

The second category of examples are about maintaining a global formation while satisfying a set of reach-avoid specifications. We show how the formation control problem can be expressed using a static obstacle $Avoid$ on the product state space $X^\times$.

Let us first formalize the notion of *formation*. Let $\{ \Sigma^i = (X^i, x_{\text{in}}^i, U^i, W^i, f^i) \}$ be a set of robots. A *formation constraint* is a set $\{ \lambda^{i,j} \in \mathbb{R} \}_{i,j \in [1;N]}$ where every $\lambda^{i,j}$ specifies the relative *Euclidean* distance between the projections of state of robot Σ^i and robot Σ^j.

Now suppose $Goal^i \subseteq X^i$ are the individual goal states, $Obs \subset \mathbb{R}^2$ is a common obstacle $\delta \in \mathbb{R}_{>0}$ is a safety margin, and $\mu \in \mathbb{R}_{>0}$ is a tolerance margin for the formation constraint. The formation control problem then asks to generate controllers $\{ C_i \}$ such that every robot Σ^i eventually reaches $Goal^i$ while avoiding Obs by the margin δ, as well as while making sure that the *Euclidean* distance between robots Σ^i and Σ^j is in the range $\lambda^{i,j} \pm \mu$. Essentially the tolerance margin μ is to account for the possible slight deviations due to disturbances experienced by the robots. Notice that since the robots have their own goals but at the same time they need to "stay close" to their neighboring robots in the formation for the entire period, they might first need to accompany the other robots to their goals, before being accompanied by them to reach their own goal. We can express the formation control problem in the product state space as follows:

- The goal set $Goal \subseteq X^{\times}$ is defined as $Goal := Goal^1 \times \ldots \times Goal^N$, and
- The avoid set $Avoid \subseteq X^{\times}$ is

$$\left\{ x^{\times} \in X^{\times} \;\middle|\; \begin{array}{c} \exists i \in [1;N] \;.\; D(x^i, Obs) \leq \delta \\ \vee \\ \exists i,j \in [1;N] \;.\; i \neq j \;.\; d\left(x^i, x^j\right) \notin \lambda^{i,j} \pm \mu \end{array} \right\}, \quad (3)$$

where the last disjunction in the definition of $Avoid$ is the restriction required for maintaining the formation, and the rest are same as in Sec. 4.1.

4.2.1 Multi-drone formation control. Consider a formation control scenario where a set of five drones (identically modeled) need to go from a specified start point to a certain destination (both defined over the corresponding state spaces) within a finite horizon, while four of them forming a diamond around a fifth drone (positioned at the diamond's center) at every time point. There are two square-shaped obstacles from which the group needs to keep a certain minimum distance at all of the time points.

The nominal dynamics for each of the drones is the same as the one given in Sec. 4.1.1. The disturbance set and robustness margin are chosen as $|W| \leq \begin{bmatrix}0.03 & 0.03 & 0.03\end{bmatrix}^T$ and $\varepsilon = \begin{bmatrix}0.18 & 0.18 & 0.18\end{bmatrix}^T$. Distance between each pair of drones positioned at the diamond's vertices is set to be $\lambda^{i,j} = \frac{3\sqrt{2}}{2}$ for $i,j \in \{1,2,3,4\}$, while the drone positioned at the center is supposed to keep distance $\lambda^{5,j} = 1.5$ for $j \in \{1,2,3,4\}$. Setting the minimum distance for obstacle avoidance to $\delta = 0.4$ and horizon length $T = 100$, ALTRO finds a valid solution over the product system with 15 state and 10 input variables within 163 seconds. Next, we synthesize local controllers for every drone such that the specifications hold for the perturbed models with $\mu = 0.5$. We consider state and input spaces to be $X = [-2,17] \times [-2,17] \times [0.6,1.6]$ and $U = [-0.9,4.8] \times [-3,3]$, respectively. We select $\eta_X = \begin{bmatrix}0.02 & 0.02 & 0.02\end{bmatrix}^T$ and $\eta_U = \begin{bmatrix}0.3 & 0.15\end{bmatrix}^T$. Tab. 1 illustrates list of run times and number of input-state pairs corresponding to local and global ABCD. When the number of agents $N > 1$, the memory requirement for global ABCD exceeds our limits (1.5TB of RAM). Fig. 8 illustrates four sequential frames of a sample trajectory generated by employing feedback controllers. Notice that both relative position and orientation between drones are kept (almost) constant throughout the journey. On the other hand, using ALTRO alone would not provide guarantee against bounded disturbance. Fig. 9 illustrates performance of open-loop and feedback controllers on regulating distance between two of the drones with and without disturbance. As expected, in the absence of disturbance open-loop controller suffices and distance between the two drones (shown in solid blue) does not go below the threshold line. However, when constant additive disturbance vectors $\begin{bmatrix}0 & 0.03 & 0.03\end{bmatrix}^T$ and $\begin{bmatrix}0 & -0.03 & -0.03\end{bmatrix}^T$ are being applied to the two drones throughout the whole horizon, open-loop controller fails, whereas the feedback controller is still capable of maintaining distance above the given threshold.

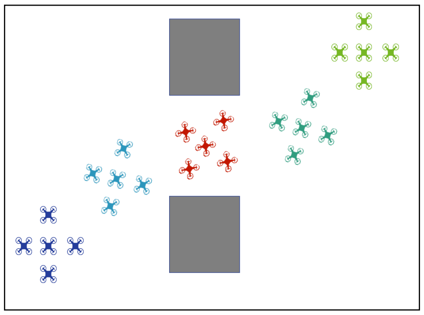


Figure 8: Illustration of a sample trajectory generated by the feedback controllers for the formation control example

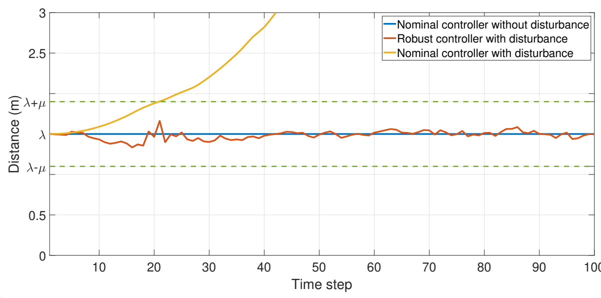


Figure 9: Comparison of open-loop and feedback controllers for the formation control example

5 DISCUSSION

MS: We have presented a decentralized controller synthesis scheme for multi-agent systems with global reach-avoid specifications. In this section, we highlight the features of our work, remaining challenges and the future work.

Scalability of the proposed method. We compute the reference trajectory for the product system using ALTRO. Our experiments show that ALTRO generates reference trajectories for high-dimension systems quite fast. On the other hand, for tracking controllers, our method synthesizes controllers in a decentralized manner and thus scalability is not affected by increasing the number of agents. We use SCOTS for computing tracking controllers because the algorithm can be effectively parallelized [18]. While our tool uses ALTRO and SCOTS for solving the given reach-avoid problem, we emphasize that one could replace them with any other off-the-shelf method that can perform similar tasks.

Comparison to other tools that are able to solve reach-avoid specifications. Our experiments show that our method is efficient in solving reach-avoid tasks for multi-agent systems. However, there are other tools which claim they can solve similar tasks. Table 2 lists main features for some of such active tools.

Table 2: Features of different tools.

Tool	Method	Dynamics	Type of disturbance
Ours(??)	decentralized	non-linear	bounded additive
FastTrack	centralized	non-linear	-
Factest	??	??	??
RTD	??	??	??

Extension to richer classes of specification. Our proposed method is specific to reach-avoid specifications. However, application of our method can be extended to the problem that can be broken into a sequence of reach-avoid tasks. To that end, one can use high-level languages (e.g., [13, 22]) for specifying complex reach-avoid sequences and invoke our tool to solve each individual reach-avoid task.

Choice of parameters.

Conservativeness of our method.

REFERENCES

[1] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43(5):1079–1100, 2019.

[2] Rajeev Alur, Salar Moarref, and Ufuk Topcu. Pattern-based refinement of assume-guarantee specifications in reactive synthesis. In *TACAS*, pages 501–516. Springer, 2015.

[3] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[4] S. Bansal, M. Chen, J. F. Fisac, and C. J. Tomlin. Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information. In *ACC*, 2017.

[5] G. B. Banusic, R. Majumdar, M. Pirron, A-K. Schmuck, and D. Zufferey. PGCD: robot programming and verification with geometry, concurrency, and dynamics. In *ICCPS 2019, Montreal, QC, Canada*, pages 57–66. ACM, 2019.

[6] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE TSMCS*, SMC-13(5):834–846, September 1983.

[7] J. Chen, S. Moarref, and H. Kress-Gazit. Verifiable control of robotic swarm from high-level specifications. In *AAMAS*, pages 568–576. ACM, 2018.

[8] H.M. Choset, S. Hutchinson, K.M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[9] A. Desai, I. Saha, J. Yang, S. Qadeer, and S. A. Seshia. Drona: A framework for safe distributed mobile robotics. In *ICCPS*, pages 239–248, 2017.

[10] C. Fan, U. Mathur, S. Mitra, and M. Viswanathan. Controller synthesis made real: reach-avoid specifications and linear dynamics. In *CAV*, pages 347–366, 2018.

[11] C. Fan, K. Miller, and S. Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In *CAV*, pages 629–652, 2020.

[12] Ivan Gavran, Rupak Majumdar, and Indranil Saha. Antlab. *ACM TECS*, 16(5s):1–19, October 2017.

[13] Ritwika Ghosh, Chiao Hsieh, Sasa Misailovic, and Sayan Mitra. Koord: a language for programming and verifying distributed robotics application. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–30, November 2020.

[14] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C.J. Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *CDC*, pages 1517–1522, 2017.

[15] T.A. Howell, B.E. Jackson, and Z. Manchester. Altro: A fast solver for constrained trajectory optimization. In *IROS*, 2019.

[16] B.E. Jackson, T.A. Howell, K. Shah, M. Schwager, and Z. Manchester. Scalable cooperative transport of cable-suspended loads with uavs using distributed trajectory optimization. *IEEE Robot. Autom. Lett.*, 5(2):3368–3374, 2020.

[17] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE TRA*, 12(4):566–580, 1996.

[18] M. Khaled and M. Zamani. pfaces: an acceleration ecosystem for symbolic control. In *HSCC*, pages 252–257, 2019.

[19] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[20] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Trans. Robot. Autom.*, 14(6):912–925, 1998.

[21] Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Damien Zufferey. Assume-guarantee distributed synthesis. In *EMSOFT*, 2020.

[22] Rupak Majumdar, Nobuko Yoshida, and Damien Zufferey. Multiparty motion coordination: From choreographies to robotics programs (artifact), 2020.

[23] P.-J. Meyer, H. Yin, A.H. Brodtkorb, M. Arcak, and A. J. Sørensen. Continuous and discrete abstractions for planning, applied to ship docking. *CoRR*, abs/1911.09773, 2019.

[24] R.M. Murray, M. Rathinam, and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *Proceedings of the 1995 ASME International Congress and Exposition*, 1995.

[25] Alexandros Nikou and Dimos V. Dimarogonas. Decentralized tube-based model predictive control of uncertain nonlinear multiagent systems. *International Journal of Robust and Nonlinear Control*, 29(10):2799–2818, March 2019.

[26] P. Nilsson and A. D. Ames. Barrier functions: Bridging the gap between planning from specifications and safety-critical control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 765–772, 2018.

[27] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam. Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 186–197, 2018.

[28] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE TAC*, 62(4):1781–1796, 2016.

[29] A. Rodionova, Y. Pant, K. Jang, H. Abbas, and Rahul Mangharam. Learning-to-fly: Learning-based collision avoidance for scalable urban air mobility. *ArXiv*, abs/2006.13267, 2020.

[30] M. Rungger and M. Zamani. SCOTS: A tool for the synthesis of symbolic controllers. In *HSCC*, pages 99–104, 2016.

[31] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition.* Pearson Education, 2010.

[32] I. Saha, R. Rattanachai, V. Kumar, G. J. Pappas, and S. A. Seshia. Implan: Scalable incremental motion planning for multi-robot systems. In *ICCPS*. IEEE, April 2016.

[33] Y. E. Sahin, P. Nilsson, and N. Ozay. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In *ICCPS*, pages 249–258, 2017.

[34] S. Singh, M. Chen, S.L. Herbert, C.J. Tomlin, and M. Pavone. Robust tracking with model mismatch for fast and safe planning: an sos optimization approach. In *WAFR*, pages 545–564, 2018.

[35] M. Srinivasan, S. Coogan, and M. Egerstedt. Control of multi-agent systems with finite time control barrier certificates and temporal logic. In *CDC*, pages 1991–1996, 2018.

[36] X. Sun, R. Nambiar, M. Melhorn, Y. Shoukry, and P. Nuzzo. Dos-resilient multi-robot temporal logic motion planning. In *ICRA*, pages 6051–6057, 2019.

[37] R. Tedrake, I. R. Manchester, M. Tobenkin, and J.W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *Int. J. Robot. Res*, 29(8):1038–1052, 2010.

[38] T. Wongpiromsarn, U. Topcu, and R.M. Murray. Receding horizon temporal logic planning. *IEEE TAC*, 57(11):2817–2830, 2012.

[39] W.Xiao and C.G. Cassandras. Decentralized optimal merging control for connected and automated vehicles with optimal dynamic resequencing. In *ACC*. IEEE, July 2020.

[40] W. Xiao, C.G. Cassandras, and C. Belta. Decentralized merging control in traffic networks with noisy vehicle dynamics: a joint optimal control and barrier function approach. In *ITSC*, 2019.

[41] L. Yang and N. Ozay. Provably-correct fault tolerant control with delayed information. In *CDC*, 2017.