Route Guidance for Satisfying Temporal Logic Specifications on Aircraft Motion

Raghvendra V. Cowlagi* Zetian Zhang†

Worcester Polytechnic Institute, Worcester, MA, 01609, USA.

We present a new technique for aircraft route guidance subject to linear temporal logic (LTL) specifications. The proposed approach is based on workspace partitioning, and relies on the idea of so-called lifted graphs. Briefly, edges in a lifted graph are successions of adjacent edges in the topological graph associated with the workspace partition. We associate edges of the lifted graph with certain reachability properties of the aircraft model. The main result of this paper is the precise characterization of acceptable routes (namely, sequences of cells) that are guaranteed to be traversable by admissible state trajectories of the aircraft model while satisfying the given LTL specifications. The proposed approach incorporates nonholonomic kinematic constraints, does not require complete controllability in the presence of workspace constraints, and does not require linearization of the aircraft model. We discuss numerical methods to implement the proposed route-planning algorithm. We illustrate the proposed algorithm with numerical simulation examples that reflect the practical significance of LTL specifications in aircraft guidance.

*Assistant Professor, Aerospace Engineering Program. AIAA Member.

†Graduate Research Assistant, Aerospace Engineering Program.

I. Introduction

The demand for higher degrees of autonomy in unmanned aerial vehicles (UAVs) widens the scope of onboard guidance from the traditional task of optimal waypoint navigation to higher-level tasks involving intelligent decision-making. We discuss the role of linear temporal logic (LTL) specifications in formulating such higher-level tasks, and report a new guidance algorithm capable of satisfying LTL specifications on aircraft motion.

LTL is a formal system, similar to the commonly used propositional logic system. In addition to the standard operators `and`, `or`, and `not`, LTL includes temporal operators such as `always`, `eventually`, and `until`. LTL has been used in software design for the specification of "correct" behaviors of algorithms.[1,2] Software algorithms can be examined to determine if they meet given LTL specifications, using so-called *formal verification* methods such as model checking.[3]

More recently, LTL has been applied for specifying behaviors of dynamical systems, and in particular, behaviors of mobile vehicles.[4–6] For robotic vehicles such as UAVs, high-level intelligent "tasks" as well as properties of safe behaviors, e.g. *"perform persistent surveillance in region A until a target is found, then report data to region B, never fly in region C, and finally return to base"* can be formulated with LTL specifications. Guidance and control algorithms are then required to plan and execute UAV motions to satisfy these specifications. The state-of-the-art approaches to control subject to LTL specifications do not suffice to address the needs of aircraft kinematic and dynamic constraints (details follow). The focus of this paper is a new method for aircraft guidance under LTL specifications.

Relation to the State-of-the-Art: Several results on the control of dynamical systems to satisfy LTL specifications are reported in the literature. All of these results crucially rely on generating a *discrete abstraction* of a dynamical system,[7,8] which is a finite state transition system whose transition sequences are equivalent – via appropriately defined equivalence relations – to admissible state trajectories of the dynamical system. Discrete abstractions allow the application of formal verification and/or search algorithms to find transition sequences that satisfy the given LTL specifications. To this end, a compact domain of interest in the state space of the dynamical system Γ is partitioned into a finite number of regions. Each region in this partition is uniquely associated with a state of a transition

system $\mathcal{G}$. Control laws are designed to steer trajectories of Γ between these regions and thereby emulate state transitions in $\mathcal{G}$. The LTL specification is represented by a *Büchi automaton* $\mathcal{B}$, which is a finite state transition system with transition sequences exactly equal to those admissible under the given LTL specification.[9–12] Finally, a *product transition system* of $\mathcal{B}$ and $\mathcal{G}$ is searched. Any transition sequence of this product system can be projected to a path in $\mathcal{G}$, which in turn can be associated with control laws and admissible state trajectories of Γ. These state trajectories are therefore guaranteed to satisfy the given LTL specification.

The preceding approach is in general beneficial, and is considered "canonical" to the extent that many works in the literature assume a priori the existence of a finite transition system that represents an underlying dynamical system.[13–15] However, there are several serious shortcomings in the state-of-the-art, which the proposed work seeks to address.

First, the efficient construction of discrete abstractions is largely restricted to low-dimensional linear systems.[16–18] The approach of state-space partitioning naturally incurs the "curse of dimensionality", and leads to an explosion in the number of states in $\mathcal{G}$. The situation is worse for nonlinear systems, where the requirement of finding local control laws to steer state trajectories between adjacent regions in the aforesaid partition leads to enormous numbers of states in $\mathcal{G}$, even for low-dimensional systems. Consider for example a recently reported work,[19] where the discrete abstraction of a three-state vehicle model consists of 91,035 states and over 34 *million* transitions. Whereas the size of the discrete abstraction may not be of high relevance for offline verification purposes, it can easily overwhelm the limited computational resources onboard UAVs. Several other recent works address the satisfaction of LTL specifications for nonlinear systems, but either involve linearization,[20] or rely on strong controllability properties or special vector field structure of Γ[21,22] that are not applicable to the aircraft model considered in this paper.

Second, the state-of-the-art methods for satisfying LTL specifications do not "adapt" to changes in model parameters that affect motion characteristics, e.g. bounds on the steering rate. When such parameters do change, the entire process of computing the discrete abstraction and product transition system must be repeated, which can be computationally expensive or prohibitive.

Third, the state-of-the-art methods do not accommodate standard trajectory optimiza-

tion algorithms, which have been established for aircraft guidance.[23] This lack of context to traditional aircraft guidance algorithms therefore restricts the scope of practical application of these methods.

In this paper, we propose a novel and computationally efficient approach to aircraft guidance subject to LTL specifications, where nonholonomic constraints on the aircraft's motion are considered. The scope of applications of this work primarily involves small-scale UAVs in domains such as urban environments where the dimensions of the UAV maneuvering characteristics (e.g. minimum turn radius) are comparable to the dimensions of workspace features (e.g. distances between obstacles). The proposed approach is based on workspace partitioning, and relies on the idea of *lifted graphs*.[24] Briefly, edges in a lifted graph are successions of adjacent edges in the topological graph associated with the workspace partition. We associate edges of the lifted graph with reachability properties of the aircraft model. A finite state transition system is then constructed as the product of the lifted graph with the Büchi automaton associated with the given LTL specification. Each run of this product transition system has a unique projection on the collection of paths in the workspace partition graph. We show that each run of the product transition system is associated with an admissible state trajectory of the aircraft model that satisfies the given LTL specifications and the proposed guidance algorithm relies on searching the product transition system.

The contributions of this paper are as follows.

First, we provide a new guidance algorithm to find trajectories satisfying LTL specifications for an aircraft model with nonholonomic motion constraints. The state-of-the-art in this area involves linearization or feedback linearization (in the absence of state- or input constraints), followed by discrete abstraction of the linearized model. The proposed algorithm does not involve linearization. The proposed algorithm relies on partitioning the *output space* (instead of the state space), which reduces the number of states and transitions in the aforesaid product transition system. Previous attempts at using output space decompositions[4] have either ignored state- and control input constraints and/or rely on strong controllability properties of the dynamical system such as controllability in the presence of workspace constraints,[21] which is not true of the aircraft model considered in this paper.

Second, we emphasize offline preprocessing of a significant portion of the computations

in the proposed route-planning algorithm, thereby elevating the practical applicability of the proposed approach in future UAV onboard, real-time guidance systems.

Third, the proposed algorithm accommodates independent trajectory optimization algorithms for generating reference trajectories. Also, in contrast to the state-of-the-art,[4,17,25] the proposed approach does not use up control authority for the sake of creating a discrete abstraction of the vehicle model. Furthermore, we discuss a local trajectory generation problem which can be solved by standard numerical trajectory optimization tools. These two features of the proposed approach not only afford the flexibility of using the proposed algorithm in conjunction with existing trajectory optimization algorithms for UAVs, but also allow the use of the full range of a UAV's maneuvering capabilities.

Fourth, the proposed approach accommodates changes in model parameters that affect motion characteristics. Specifically, the proposed approach affords the capability of making incremental changes in an existing solution, in response to changes in such model parameters. This capability is afforded by the structure of lifted graphs, (i.e. it is independent of the aforesaid product operation with the Büchi automaton). A complete description and analysis of an algorithm for making such incremental changes is beyond the scope of this paper, and we refer the reader to our preliminary work in this context[26] instead.

The rest of this paper is organized as follows. In Section II, we precisely formulate the problem. In Section III, we discuss the lifted graph and a discrete abstraction of the vehicle model. Edge transitions in the lifted graph are discussed in context with the reachability properties of the vehicle model. In Section IV, we discuss a product transition system that enables the search for admissible trajectories satisfying the given LTL specification. In Section V, we discuss the numerical computations of edge transition costs in the lifted graph. In Section VI, we present illustrative examples of implementation of the proposed approach. We conclude the paper in Section VII with comments on future work.

II. Problem Formulation

We introduce the following elements of the problem: the vehicle model, the workspace partition, and the LTL specification. We use the term *trajectory* to refer to a locus of points in the state space of the vehicle model, and the term *route* to refer to a discrete

representation of a trajectory (e.g., a sequence of waypoints or regions). We discuss *route-planning* (synonymous with *route guidance*), which is the process of finding a sequence of regions in the workspace that enclose the aircraft's desired trajectory.

VEHICLE MODEL: Let $\mathcal{W} \subset \mathbb{R}^2$ be a compact set, called the *workspace*, which is assumed to be a planar region of interest for the vehicle, and $\nu_{\max} > \nu_{\min} > 0$ indicate, respectively, upper and lower bounds on the vehicle speed. Let $\xi = (x, y, \nu, \psi) \in \mathcal{D} = \mathcal{W} \times [\nu_{\min}, \nu_{\max}] \times \mathbb{S}^1$ denote the state of the vehicle, namely, the position of the vehicle's center of mass and the magnitude and direction of its velocity vector in a prespecified Cartesian coordinate system. We denote by $\mathsf{x}(\xi)$ the projection of $\xi \in \mathcal{D}$ on the set $\mathcal{W}$. We consider a vehicle kinematic model described by the differential equations

$$\dot{x}(t) = \nu(t)\cos\psi(t), \qquad \dot{y}(t) = \nu(t)\sin\psi(t), \qquad \dot{\nu}(t) = u_1(t), \qquad \dot{\psi}(t) = \frac{u_2(t)}{\nu(t)}, \tag{1}$$

where u_1 and u_2 (the tangential and lateral accelerations, respectively) are the control inputs. We assume that the set of admissible control input values is the compact domain

$$U := \left\{ (u_1, u_2) \in \mathbb{R}^2 \mid \frac{u_1^2}{a^2} + u_2^2\rho^2 \leqslant 1 \right\}, \tag{2}$$

where $a, \rho > 0$ are prespecified. Let $\mathcal{U}$ be the set of all piecewise continuous functions of t defined on finite intervals that take values in U. For any $u \in \mathcal{U}$, and initial state $\xi_0 \in \mathcal{D}$, the state trajectory $\xi(t; \xi_0, u)$, $t \in [0, t_{\mathrm{f}}]$, obtained by integrating (1) is called an *admissible state trajectory*. Note that a is an upper bound on the magnitude of the tangential acceleration, and ρ is the minimum radius of turn at unit speed. For computational reasons to be clarified later, we assume $\rho\nu_{\min} > 3$.

WORKSPACE PARTITION: Consider a partition of $\mathcal{W}$ into convex polytopic subregions called *cells*. The intersection of any two cells is either empty, or a single vertex, or a finite-length segment that lies on the boundaries of both cells. We denote by $N^{\mathrm{C}} \in \mathbb{Z}_+$ the number of cells, and by $R^i \subset \mathcal{W}$ the subregion associated with the i^{th} cell, for each $i = 1, \ldots, N^{\mathrm{C}}$. Therefore, $\cup_{i=1}^{N^{\mathrm{C}}} R^i = \mathcal{W}$. We associate with this partition an undirected graph $\mathcal{G} := (V, E)$ such that each vertex of $\mathcal{G}$ is uniquely associated with a cell, and each edge of $\mathcal{G}$ is uniquely associated with a pair of geometrically adjacent cells. We assume "4-connectivity," i.e., two cells are considered geometrically adjacent if their intersection is a finite-length segment. We

denote by $\mathsf{cell}(v)$ the element of $\{R^i\}_{i=1}^{N^{\mathrm{C}}}$ associated with the vertex $v \in V$. A *path* $\mathbf{v}$ in $\mathcal{G}$ is a sequence $(v_0, v_1, \ldots)$ of vertices, such that $v_0, v_k \in V$, and $(v_{k-1}, v_k) \in E$, for each $k \in \mathbb{N}$. The number of vertices in a path is called its *length*. According to the preceding definition, a path in $\mathcal{G}$ can contain cycles. We denote by $\mathcal{L}_{\mathcal{G}}$ the collection of all paths in $\mathcal{G}$. Note that the paths in $\mathcal{L}_{\mathcal{G}}$ are associated with vehicle routes described by a sequence of successively adjacent cells.

For every $t_{\mathrm{f}} \in \mathbb{R}_+$, $\xi_0 \in \mathcal{D}$, and $u \in \mathcal{U}$, we define the $\mathcal{G}$-*trace* of the trajectory $\xi(t; \xi_0, u)$, $t \in [0, t_{\mathrm{f}}]$ as the path $tr(\xi, \mathcal{G}) = (v_0, v_1, \ldots, v_P) \in \mathcal{L}_{\mathcal{G}}$ of minimal length such that

1. $\mathsf{x}(\xi(0; \xi_0, u)) \in \mathsf{cell}(v_0)$,
2. There exists a positive and strictly increasing sequence $\{t_0, t_1, \ldots, t_P\}$with $t_0 = 0$, $t_P = t_{\mathrm{f}}$, and

$$\mathsf{x}(\xi(t; \xi_0, u)) \in \mathsf{cell}(v_k), \qquad t \in [t_{k-1}, t_k], \qquad \text{for each } k = 1, 2, \ldots, P. \tag{3}$$

The preceding definition applies also to trajectories defined over $[0, \infty)$, where the aforesaid increasing sequence is infinite, and the $\mathcal{G}$-trace is infinitely long. We denote by $\mathcal{L}_\Gamma(\xi_0) \subseteq \mathcal{L}_{\mathcal{G}}$ the collection of $\mathcal{G}$-traces of all admissible trajectories for every $t_{\mathrm{f}} \in \mathbb{R}_+$, including those defined over $[0, \infty)$. Informally, the path $tr(\xi, \mathcal{G})$ is associated with the sequence of cells that defines a "channel" in $\mathcal{W}$, such that the curve $\mathsf{x}(\xi(t))$, $t \in [0, t_{\mathrm{f}}]$, lies within this channel. The curve $\mathsf{x}(\xi(t))$ and the trajectory $\xi(t)$ are said to *traverse* this channel of cells.

LTL$_{-X}$ SPECIFICATIONS: Linear temporal logic is a convenient formal language to express specifications on the behavior of a system over time. As is common in the literature,[17] we use a restricted version of LTL, namely, LTL$_{-X}$, which does not involve the *next* operator. The choice of LTL$_{-X}$ instead of LTL is for simplicity of exposition of the proposed work. A brief overview of LTL$_{-X}$ is provided below; the reader is referred to the literature[2,17] for further details.

The LTL$_{-X}$ syntax involves operators $\neg$ (negation), $\vee$ (disjunction) and $\triangleright$ (*until*). Let $\Lambda = \{\lambda_k\}_{i=0}^{N^{\mathrm{R}}}$, with $N^{\mathrm{R}} \in \mathbb{Z}_{\geqslant 0}$ be a prespecified set of atomic propositions. A *LTL$_{-X}$ formula over* Λ is recursively defined as follows:

1. Every atomic proposition $\lambda_k \in \Lambda$ is a LTL$_{-X}$ formula.

2. If ϕ_1 and ϕ_2 are LTL$_{-X}$ formulae, then $\neg\phi_1$, $(\phi_1 \vee \phi_2)$, and $(\phi_1 \triangleright \phi_2)$ are also LTL$_{-X}$ formulae.

Let ϕ_1 and ϕ_2 be LTL$_{-X}$ formulae. The formula $(\phi_1 \triangleright \phi_2)$ means that ϕ_2 eventually becomes true and ϕ_1 remains true until ϕ_2 becomes true. The operators $\wedge$ (conjunction), $\Rightarrow$ (implication), $\Leftrightarrow$ (equivalence) are defined as is standard in propositional logic. The temporal operators $\Diamond$ (*eventually*), and $\Box$ (*always*) are defined as follows:

$$\Diamond\phi_1 := (\phi_1 \vee \neg\phi_1) \triangleright \phi_1, \qquad\qquad \Box\phi_1 := \neg(\Diamond\neg\phi_1).$$

A *word* $\overline{\omega} = (\omega_0, \omega_1, \ldots)$ is a sequence such that $\omega_i \in 2^\Lambda$ for each $i = 0, 1, \ldots$ where 2^Λ denotes the power set of Λ. For $i, j \in \mathbb{Z}_{\geqslant 0}$, $j \geqslant i$, we denote by ω_i^j the word $(\omega_i, \omega_{i+1}, \ldots, \omega_j)$. The *satisfaction* of a LTL$_{-X}$ formula ϕ by the word $\overline{\omega}$ is denoted by $\overline{\omega} \models \phi$, and it is recursively defined as follows:

1. $\omega \models \lambda_k$ if $\lambda_k \in \omega_0$, and $\omega \not\models \lambda_k$ if $\lambda_k \notin \omega_0$.
2. $\omega \models \neg\phi$ if $\omega \not\models \phi$.
3. $\omega \models (\phi_1 \vee \phi_2)$ if $\omega \models \phi_1$ or $\omega \models \phi_2$.
4. $\omega \models (\phi_1 \triangleright \phi_2)$ if there exists $i \geqslant 0$ such that $\omega_i^\infty \models \phi_2$ and for every $j < i$, $\omega_j^i \models \phi_1$.

We assume a finite number of regions of interest in the workspace labeled as $\lambda_1, \ldots, \lambda_{N^{\mathrm{R}}}$. Each λ_k is an atomic proposition defined by a set membership relation in $\mathcal{D}$ of the form

$$\lambda_k \equiv \quad \mathsf{x}(\xi) \in \cup_{i \in \varsigma_k} R^i, \quad \text{for each } k = 1, \ldots, N^{\mathrm{R}}. \tag{4}$$

where $\varsigma_k \subseteq \{1, \ldots, N^{\mathrm{C}}\}$ is prespecified for each $k = 1, \ldots, N^{\mathrm{R}}$. Each region of interest is assumed to be a (possibly disconnected) union of cells. Each path $\mathbf{v} = (v_0, v_1, \ldots) \in \mathcal{L}_\mathcal{G}$ defines a word $\overline{\omega}(\mathbf{v}) = (\omega_0, \omega_1, \ldots,)$, where

$$\omega_\ell := \left\{ \lambda_k \mid \mathsf{cell}(v_\ell) \subseteq \cup_{i \in \varsigma_k} R^i \right\}. \tag{5}$$

The path $\mathbf{v}$ is said to satisfy a LTL$_{-X}$ formula ϕ if $\overline{\omega}(\mathbf{v}) \models \phi$. The main problem of interest in this paper is the following.

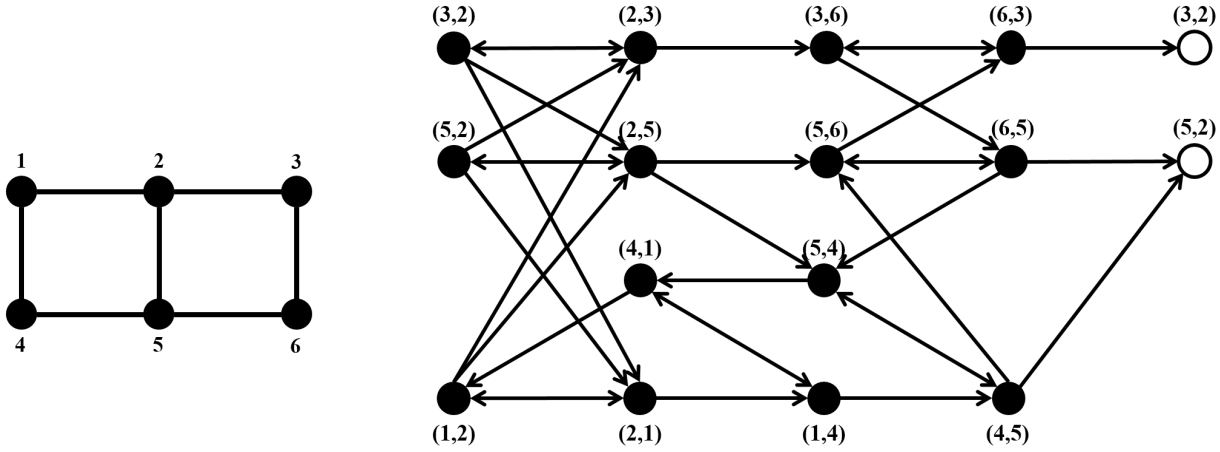
Problem 1. *Given a LTL$_{-X}$ formula ϕ over Λ, and $\xi_0 \in \mathcal{D}$, determine a collection of paths $\mathcal{L}_{\Gamma\phi} \subseteq \mathcal{L}_\Gamma(\xi_0)$ such that every path in $\mathcal{L}_{\Gamma\phi}$ satisfies the formula ϕ.*

The "channel" of cells associated with every path in the collection $\mathcal{L}_{\Gamma\phi}$ can be traversed by an admissible state trajectory. Furthermore, every path in $\mathcal{L}_{\Gamma\phi}$ also satisfies the specification ϕ. The collection $\mathcal{L}_{\Gamma\phi}$ therefore represents, loosely speaking, an equivalence class of admissible state trajectories that satisfy the specification ϕ. The computation of $\mathcal{L}_{\Gamma\phi}$ is desirable because every route that belongs to $\mathcal{L}_{\Gamma\phi}$ is guaranteed to be compatible with vehicle dynamical constraints. As we discuss in Section IV, the solution of Problem 1 immediately leads to a route-planning algorithm.

The computation of $\mathcal{L}_{\Gamma\phi}$ is challenging because $\mathcal{L}_{\Gamma}(\xi_0)$ is difficult to compute, and involves discrete abstraction of the continuous system Γ. Previously,[4,21] feedback control laws have been designed to construct a language equivalent discrete abstraction, such that $\mathcal{L}_{\Gamma} = \mathcal{L}_{\mathcal{G}}$. However, the underlying assumption therein is that the vehicle model is completely controllable in the presence of workspace constraints (e.g. obstacles). For the vehicle model considered in this paper, this controllability assumption is not true.[24]

To solve Problem 1 in the light of the preceding observations, we propose a new approach based on the so-called lifted graph, which we discuss next.

III. Lifted Graph



(a) Graph $\mathcal{G}$.

(b) Lifted graph $\mathcal{G}_1$.

Figure 1. Example of a lifted graph (the vertices $(3,2)$ and $(5,2)$ of the lifted graph are drawn twice for clarity of the diagram).

The proposed approach to the solution of Problem 1 and route-planning relies on traversabil-

ity analysis and assignment of transition costs to successions of edges in the graph $\mathcal{G}$. To this end, we adopt and modify the idea of lifted graphs[24] as follows. For $H \in \mathbb{Z}_{\geqslant 0}$, let

$$V_H := \Big\{(v_0, \ldots, v_H) : (v_{k-1}, v_k) \in E \text{ and } v_k \neq v_m \Leftrightarrow k \neq m, \text{ for } k, m \in \{1, \ldots, H\}\Big\}.$$

Every element $\mathbf{i} \in V_H$ is an ordered $(H+1)$-tuple of unique elements of V. Per the notation used by Cowlagi & Tsiotras,[24] we denote by $[\mathbf{i}]_k$ the k^{th} element of $\mathbf{i}$, and by $[\mathbf{i}]_k^m$ the tuple $\big([\mathbf{i}]_k, [\mathbf{i}]_{k+1}, \ldots, [\mathbf{i}]_m\big)$, where $1 \leqslant k < m \leqslant H+1$. Let E_H be a set of pairs $(\mathbf{i}, \mathbf{j}) \in V_H \times V_H$, such that $[\mathbf{i}]_k = [\mathbf{j}]_{k-1}$, for each $k = 2, \ldots, H+1$. According to this notation, $V_0 = V$ and $E_0 = E$. Note that, for $H \geqslant 1$, each vertex in V_H defines an *edge* in E_{H-1}. if H is larger than the length of the longest simple path in $\mathcal{G}$, then V_H is empty. The *lifted graph* $\mathcal{G}_H$ is defined as the directed graph whose vertex and edge sets are, respectively, V_H and E_H. Figure 1 illustrates an example of a lifted graph for $H = 1$.

With the exception of paths that contain cycles of length $H+1$ or less, every path $\mathbf{v} = (v_0, v_1, \ldots)$ in the graph $\mathcal{G}$ can be uniquely mapped to a path $\bar{\mathbf{i}} := (\mathbf{i}_0, \mathbf{i}_1, \ldots)$ in the lifted graph $\mathcal{G}_H$, where $\mathbf{i}_k := (j_k, j_{k+1}, \ldots, j_{k+H}) \in V_H$ for each $k \in \mathbb{N}$. We denote this map by $b_0^H : \mathcal{L}_\mathcal{G} \to \mathcal{L}_{\mathcal{G}_H}$, where $\mathcal{L}_{\mathcal{G}_H}$ is the collection of all paths in $\mathcal{G}_H$. Therefore, $\bar{\mathbf{i}} = b_0^H(\mathbf{v})$ and $\mathbf{v} = b_H^0(\bar{\mathbf{i}})$. For every integer $H \geqslant 0$, the map b_0^H is surjective and invertible and b_H^H is the identity map.

As a first step towards the solution of Problem 1, we associate certain reachability properties of the vehicle model with edge transitions in the lifted graph $\mathcal{G}_H$. In what follows, we assume $H \geqslant 1$.

III-A. Edge Transition Costs

To characterize the collection $\mathcal{L}_\Gamma$, we define a transition cost function $g_H : E_H \to [0, \chi]$, by which we can compute costs of paths in the lifted graph $\mathcal{G}_H$. Here, $\chi \in \mathbb{R}_+$ is a large positive number sufficiently greater than the length of the longest simplest path in $\mathcal{G}$. Then we assert that a path $\mathbf{v} \in \mathcal{L}_\mathcal{G}$ belongs to the collection $\mathcal{L}_\Gamma(\xi_0)$ if and only if any subpath of $b_0^H(\mathbf{v})$ of length less than χ has cost less than χ.

Consider an edge $(\mathbf{i}, \mathbf{j}) \in E_H$ in the lifted graph $\mathcal{G}_H$. Let $\mathcal{S}(\mathbf{i}) \subset \mathcal{D}$ be a set of states associated with $\mathbf{i} \in V_H$ such that $\mathsf{x}(\mathcal{S}(\mathbf{i})) \subseteq \mathsf{cell}([\mathbf{i}]_1) \cap \mathsf{cell}([\mathbf{i}]_2)$. Thus, the projections on $\mathcal{W}$ of the elements in $\mathcal{S}(\mathbf{i})$ lie on the boundary between the first and second cells corresponding

to the vertices of V that constitute the ordered H-tuple $\mathbf{i}$. Next let $\mathcal{Q}(\mathbf{j}) \subset \mathcal{D}$ be such that $\mathsf{x}(\mathcal{Q}(\mathbf{j})) \subseteq \mathsf{cell}([\mathbf{j}]_1) \cap \mathsf{cell}([\mathbf{j}]_2)$ and for every $\xi_\mathrm{q} \in \mathcal{Q}(\mathbf{j})$ there exists a finite traversal time t_q and an admissible control input $u_\mathrm{q} \in \mathcal{U}$ such that $tr(\xi(\cdot; \xi_\mathrm{q}, u_\mathrm{q}), \mathcal{G}) = b^0_{H-1}(\mathbf{j})$. Informally, $\mathcal{Q}(\mathbf{j})$ is the set of all states whose position components lie on the boundary between the first and second cells of $\mathbf{j}$, and such that the traversal of the geometric region defined by the cells associated with $\mathbf{j}$ is possible from any initial state within $\mathcal{Q}(\mathbf{j})$. Sets such as $\mathcal{Q}(\cdot)$ are called *backward reachable sets* or *target sets*.[27]

Next, let $\mathcal{R}_\mathbf{i}$ be a reachability map associated with the states in sets $\mathcal{S}(\mathbf{i})$, defined by

$$\mathcal{R}_\mathbf{i}(\xi_\mathrm{s}) := \left\{ \xi_\mathrm{t} \in \mathcal{D} \mid \xi_\mathrm{t} \in \bigcup_{t \in \mathbb{R}_+} \bigcup_{u \in \mathcal{U}_t} \xi(t; \xi_\mathrm{s}, u), \text{ and } \left(\bigcup_{\tau \in [0,t]} \mathsf{x}(\xi(\tau; \xi_\mathrm{s}, u)) \right) \cap (\mathcal{W} \backslash \mathsf{cell}([\mathbf{i}]_2)) = \varnothing \right\}, \tag{6}$$

where $\xi_\mathrm{s} \in \mathcal{S}(\mathbf{i})$. Informally, $\mathcal{R}_\mathbf{i}(\xi_\mathrm{s})$ is a *forward reachable set* of all states that can be reached from ξ_s by trajectories whose projections on $\mathcal{W}$ always remain within the $\mathsf{cell}([\mathbf{i}]_2)$, i.e. the region defined by the second cell of $\mathbf{i}$. Finally, define $\hat{\mathcal{S}}(\mathbf{i}, \mathbf{j}) := \{\xi_\mathrm{s} \in \mathcal{S}(\mathbf{i}) : \mathcal{R}_\mathbf{i}(\xi_\mathrm{s}) \cap \mathcal{Q}(\mathbf{j}) \neq \varnothing\}$.

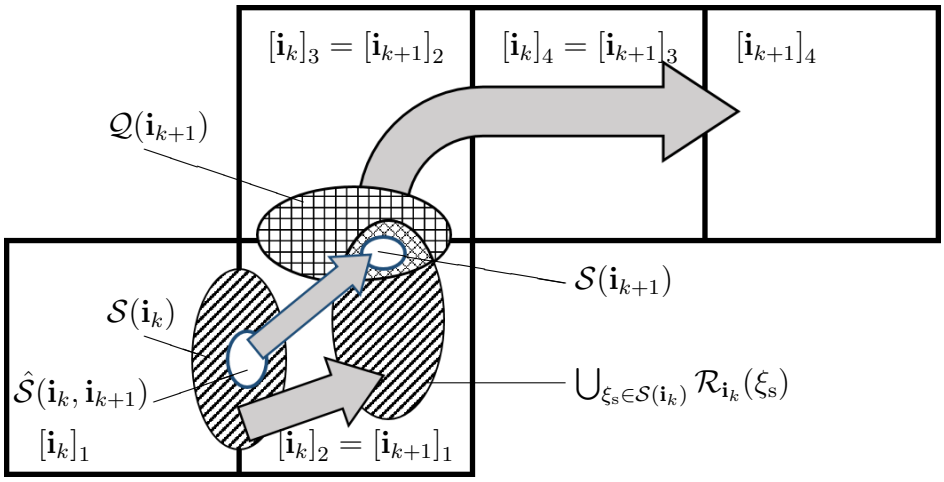


Figure 2. Conceptual illustration of the sets $\mathcal{R}$, $\mathcal{Q}(\cdot)$ and $\mathcal{S}(\cdot)$ for $H = 2$. The large arrows indicate the existence of admissible state trajectories as described in the text preceding (6).

Now consider a finite-length path $\mathbf{v} = (v_0, v_1, \ldots v_P) \in \mathcal{L}_\mathcal{G}$ with no cycles of length less than or equal to $H + 1$, and an initial vehicle state $\xi_0 \in \mathcal{D}$, with $\mathsf{x}(\xi_0) \in \mathsf{cell}(v_0) \cap \mathsf{cell}(v_1)$. For each $k \in \mathbb{N}$, let $\mathbf{i}_k := (v_k, \ldots, v_{k+H})$. Clearly, $(\mathbf{i}_k, \mathbf{i}_{k+1}) \in E_H$. We iteratively define g_H

and the set association $\mathcal{S}(\cdot)$ as follows:

$$\mathcal{S}(\mathbf{i}_{k+1}) := \bigcup_{\xi_s \in \hat{\mathcal{S}}(\mathbf{i}_k, \mathbf{i}_{k+1})} \left(\mathcal{R}_{\mathbf{i}_k}(\xi_s) \cap \mathcal{Q}(\mathbf{i}_{k+1})\right), \tag{7}$$

$$g_H(\mathbf{i}_k, \mathbf{i}_{k+1}) := \begin{cases} \chi, & \text{if } \mathcal{S}(\mathbf{i}_{k+1}) = \varnothing, \\ 1, & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathcal{S}(\mathbf{i}_0) = \xi_0$. The *H-cost* of the path $\mathbf{v} \in \mathcal{L}_\mathcal{G}$ is defined as

$$\mathcal{J}_H(\mathbf{v}) := H + \sum_{k=0}^{P-H} g_H(\mathbf{i}_k, \mathbf{i}_{k+1}). \tag{9}$$

Figure 2 illustrates these concepts. Note first that the H-cost of a path $\mathbf{v} \in \mathcal{L}_\mathcal{G}$ depends on the initial state. The collection of sets $\mathcal{S}(\mathbf{i}_k)$ indicate the possible vehicle states from which traversal of the channel associated with the path $\mathbf{v}$ is possible. Second, note that due to the recursive nature of (7)–(9), it is beneficial to compute edge transition costs in $\mathcal{G}_H$ in conjunction with a search algorithm for finding a desired path in $\mathcal{L}_\mathcal{G}$.

In this context, notice that a serious issue arises. On the one hand, the computations of the sets $\mathcal{R}(\cdot)$, $\mathcal{Q}(\cdot)$, and $\mathcal{S}(\cdot)$ are time-intensive and unsuitable for real-time implementation. On the other hand, the edge transition costs in (7)–(9) must be computed simultaneously with the real-time route-planning algorithm (see Section IV) that searches for a path in the lifted graph $\mathcal{G}_H$. To resolve this issue, we propose a technique that relegates the time-intensive computations to an offline preprocessing stage, and yet allows for conservative approximations of the sets $\mathcal{R}(\cdot)$, $\mathcal{Q}(\cdot)$, and $\mathcal{S}(\cdot)$ to be determined online (based on the preprocessed results) within the route-planning algorithm. This technique is discussed in Section V. Here, we state an important result that characterizes the collection $\mathcal{L}_\Gamma(\xi_0)$ of $\mathcal{G}$-traces of all admissible vehicle trajectories.

Proposition 1. *Let $\mathbf{v} = (v_0, \ldots, v_P)$ be a path in $\mathcal{L}_\mathcal{G}$ with length less than χ and with no cycles of length less than or equal to $H+1$, and let $\xi_0 \in \mathcal{D}$ be prespecified such that $\mathsf{x}(\xi_0) \in \mathsf{cell}(v_0) \cap \mathsf{cell}(v_1)$. Then $\mathbf{v} \in \mathcal{L}_\Gamma(\xi_0)$ if and only if $\mathcal{J}_H(\mathbf{v}) < \chi$.*

Proof. See Appendix A. □

Proposition 1 asserts that the channel of cells associated with any path in $\mathcal{L}_\mathcal{G}$ is traversable by an admissible state trajectory of Γ if and only if the H-cost of this path is less than χ (assuming that the number of vertices in this path is less than χ).

IV. Product Transition System and Route-Planning Algorithm

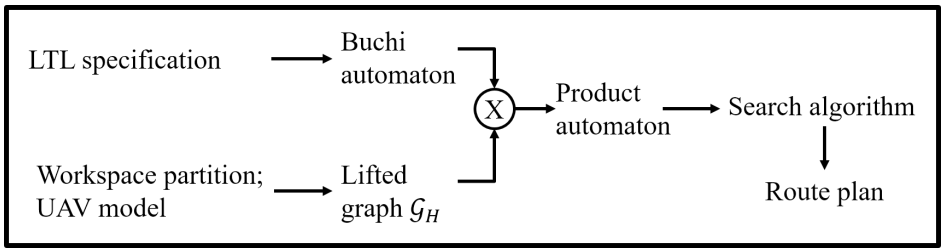


Figure 3. Conceptual illustration of the proposed route-planning algorithm.

The lifted graph $\mathcal{G}_H$ and the associated edge transition cost computations define a finite state transition system that represents the vehicle dynamical model. To find paths in $\mathcal{G}_H$ that satisfy a given specification ϕ, we construct and search a so-called *product transition system* as discussed below. Figure 3 illustrates the proposed route-planning algorithm. Recall that the connection between the given LTL specification and the UAV route is via (4), which associates propositions in the LTL specifications with regions in the workspace.

It is known[9,10] that every LTL formula can "represented" by a finite state transition system. Precisely, every LTL formula ϕ over the alphabet Λ is associated with a *Büchi automaton* $\mathcal{B}_\phi$ with input alphabet 2^Λ. The collection of accepting runs of $\mathcal{B}_\phi$ is exactly the collection of infinite strings over Λ that satisfy ϕ. In the context of UAV route-planning, admissible transition sequences in $\mathcal{B}_\phi$ are associated with routes that satisfy the given LTL specification. Algorithms for translating a LTL formula to the associated Büchi automaton are available.[10–12] The reader interested is referred to the literature[2,28] for further details on finite state automata in general and Büchi automata in particular.

For the Büchi automaton $\mathcal{B}_\phi$, we denote by S the set of states, by $\delta_{\mathcal{B}_\phi} \subseteq S \times 2^\Lambda \times S$ the transition relation between states, and by $S_0, S_\mathrm{f} \subseteq S$, respectively, the sets of initial and accepting states. For an integer $H \geqslant 1$, we now define a *product transition system* $\mathcal{T}_{\phi,H} := (T, \delta_{\mathcal{T}_{\phi,H}})$ as follows:

1. The set of states of $\mathcal{T}_{\phi,H}$ is $T := S \times V_H$. For each state $\theta \in T$, we denote by $\theta|_S$ and $\theta|_{V_H}$, respectively, the projections of θ on S and V_H.

2. The transition relation of $\mathcal{T}_{\phi,H}$ is $\delta_{\mathcal{T}_{\phi,H}} \subseteq T \times 2^\Lambda \times T$ defined as the set of all triplets

$(\theta_k, \omega_k, \theta_\ell)$ such that

$$(\theta_k|_S, \omega_k, \theta_\ell|_S) \in \delta_{\mathcal{B}_\phi}, \qquad (\theta_k|_{V_H}, \theta_\ell|_{V_H}) \in E_H, \tag{10}$$

$$\omega_k = \left\{\lambda_i \mid \mathsf{cell}([\theta_k|_{V_H}]_1) \subseteq \cup_{j\in\varsigma_i} R_j\right\}. \tag{11}$$

Thus, transitions in the product system are associated with transitions in the Büchi automaton *and* with transitions in the lifted graph $\mathcal{G}_H$. The relationship between transitions in the product system and the UAV's workspace is defined by (11).

A *run* of $\mathcal{T}_{\phi,H}$ is a sequence $\Theta = (\theta_0, \theta_1, \dots,)$ such that $\theta_k \in T$ for each $k \in \mathbb{Z}_{\geqslant 0}$, and $(\theta_k, \omega_k, \theta_{k+1}) \in \delta_{\mathcal{T}_{\phi,H}}$, with ω_k as defined in (11). We denote by $\Theta|_S = (\theta_0|_S, \theta_1|_S, \dots,)$ and $\Theta|_{V_H} = (\theta_0|_{V_H}, \theta_1|_{V_H}, \dots,)$, respectively, the projections of Θ on S and V_H. Note that $\Theta|_{V_H} \in \mathcal{L}_{\mathcal{G}_H}$, and therefore $b^0_H(\Theta|_{V_H}) \in \mathcal{L}_{\mathcal{G}}$.

Similar to previously reported approaches in the literature,[17] we restrict attention to runs of $\mathcal{T}_{\phi,H}$ of a "prefix-suffix" form $\Theta = (\Theta_{\mathrm{p}}, \Theta_{\mathrm{s}}, \Theta_{\mathrm{s}}, \dots,)$. Here, the "suffix" run $\Theta_{\mathrm{s}} = (\theta_{\mathrm{f},1}, \theta_{\mathrm{s},1}, \dots, \theta_{\mathrm{s,N}}, \theta_{\mathrm{f},2})$, which is repeated infinitely often in Θ, is a finite sequence such that $\theta_{\mathrm{f},1}, \theta_{\mathrm{f},2} \in S_{\mathrm{f}} \times V_H$, and $\theta_{\mathrm{s,n}} \in S \times V_H$, for $n = 1, \dots, N$. The "prefix" run $\Theta_{\mathrm{p}} = (\theta_0, \dots, \theta_M)$ is a finite sequence such that $\theta_0 \in S_0 \times V_H$ and $(\theta_M, \omega_M, \theta_{\mathrm{f}}) \in \delta_{\mathcal{T}_{\phi,H}}$. Now we state the main result of this paper as follows.

Theorem 1. *Let $\Theta = (\theta_0, \theta_1, \dots,)$ be a run of $\mathcal{T}_{\phi,H}$.*

$$\begin{aligned} &\text{If} \quad \mathcal{J}_H(b^0_H(\Theta_{\mathrm{p}}|_{V_H})) < \chi \quad \text{and} \quad \mathcal{J}_H(b^0_H(\Theta_{\mathrm{s}}|_{V_H})) < \chi, \\ &\text{then} \quad b^0_H(\Theta|_{V_H}) \in \mathcal{L}_{\Gamma\Phi}. \end{aligned}$$

Conversely, for every path $\mathbf{v} \in \mathcal{L}_{\Gamma\Phi}$ with no cycles of length less than or equal to $H+1$, there exists a run Θ of $\mathcal{T}_{\phi,H}$, such that $b^0_H(\Theta|_{V_H}) = \mathbf{v}$.

Proof. See Appendix A. □

IV-A. Route-planning Algorithm

Theorem 1 solves Problem 1 in that it precisely characterizes the collection $\mathcal{L}_{\Gamma\phi}$. Recall that $\mathcal{L}_{\Gamma\phi}$ is an equivalence class of admissible state trajectories that satisfy the specification ϕ. Following Theorem 1, it is easy to determine a specific plan for a given initial state $\xi_0 \in \mathcal{D}$

and a given LTL_{-X} specification ϕ. To do so, we execute Dijkstra's algorithm to search for a prefix and suffix run of the product transition system that satisfy the conditions given in Theorem 1. A straightforward algorithm for finding a run of a product automaton with minimum total number of vertices in the concatenation of prefix and suffix runs appears in the literature,[17] which we can appropriately modify for finding a run of $\mathcal{T}_{\phi,H}$. A limitation of the proposed algorithm is that routes of length less than $H+1$ or routes containing cycles of length less than or equal to $H+1$ cannot be mapped to the lifted graph $\mathcal{G}_H$, and will therefore not be found by the proposed algorithm. However, for practical applications in aircraft route-planning this fact is unlikely to restrict the proposed algorithm. The values of H are typically small integers, whereas routes to be planned in practice are over large environments. Therefore, routes with less than H cells are unlikely to be of significance. Furthermore, the aircraft's minimum radius of turn is likely to preclude short-length cycles in the route.

The only modification that needs to be made to Dijsktra's search algorithm is that the edge transition costs in the lifted graph are computed simultaneously with the search. To do so, within the search algorithm each search vertex (state of the product transition system) is associated with an index set related to the vehicle state. The exact definition of these index sets is clarified in the next Section, where we also introduce a connectivity matrix.

This idea is illustrated with pseudo-code in Fig. 4. Let $\xi_0 \in \mathcal{D}$ be the given vehicle initial state and let v^{S} be the vertex in $\mathcal{G}$ such that $\xi_0 \in \mathsf{cell}(v^{\text{S}})$. We define θ^{S} as a "dummy" start vertex in the product automaton $\mathcal{T}_{\phi,H}$ such that it is connected with zero cost to all $\theta \in T$ with $[\theta|_{V_H}]_1 = v_{\text{S}}$. As is usual in Dijkstra's algorithm,[29] each search vertex (state of the product transition system) is associated with a label d and a backpointer b. The *fringe* $\mathcal{P}$ is a set of search vertices whose label can potentially be improved. The REMOVE and INSERT functions maintain a sorting order in the fringe according to the current value of the label. The modification proposed here associates the index set R with each search vertex, which is updated in Line 8 of the pseudo-code shown in Fig. 4. The meaning and significance of the symbol C (connectivity matrix) in Line 8 will become clear in the next Section. The description in Fig. 4 is an idealization, and implementations of the proposed route-planning can be significantly sped up by using search heuristics, a discussion of which is beyond the scope of this paper. The cost function (8) is approximated as follows, to provide transition

Proposed Label-Correcting Algorithm for Route-Planning

procedure INITIALIZE(θ_{S})

1: $\mathcal{P} := \theta^{\mathrm{S}}, \quad d(\theta^{\mathrm{S}}) := 0, \quad R(\theta^{\mathrm{S}}) := m_0.$

2: **for all** $\theta \in T$ **do**

3: $\quad d(\theta) := \infty$

procedure MAIN

1: INITIALIZE(θ^{S})

2: **while** $\mathcal{P} \neq \varnothing$ **do**

3: $\quad \theta_k := \textsc{Remove}(\mathcal{P})$

4: $\quad$ **for all** $\theta_\ell \in T$ such that there exists ω_k as in (11) and $(\theta_k, \omega_k, \theta_\ell) \in \delta_{\mathcal{T}_{\phi,H}}$ **do**

5: $\quad\quad$ **if** $d(\theta_k) + \tilde{g}_H((\theta_k|_{V_H}, \theta_\ell|_{V_H})) < d(\theta_\ell)$ **then**

6: $\quad\quad\quad d(\theta_\ell) := d(\theta_k) + \tilde{g}_H((\theta_k|_{V_H}, \theta_\ell|_{V_H})); \quad b(\theta_\ell) := \theta_k$

7: $\quad\quad\quad R(\theta_\ell) := \{p \in \mathbb{N} \mid C_{mp} = 1, \text{ for each } m \in R(\theta_k)\}$

8: $\quad\quad\quad \mathcal{P} := \textsc{Insert}(\mathcal{P}, \theta_\ell)$

Figure 4. Pseudo-code for executing a modified form of Dijkstra's algorithm on the product automaton.

costs in the product system $\mathcal{T}_{\phi,H}$:

$$\tilde{g}_H((\theta_k|_{V_H}, \theta_\ell|_{V_H})) := \begin{cases} \chi, & \text{if } R(\theta_\ell) = \varnothing, \\ 1, & \text{otherwise.} \end{cases}$$

The computational complexity of the proposed algorithm is the same as that of Dijkstra's algorithm, namely, $\mathcal{O}\left(|\delta_{\mathcal{T}_{\phi,H}}| + |\mathcal{T}_{\phi,H}| \log |\mathcal{T}_{\phi,H}|\right)$ when the fringe $\mathcal{P}$ is implemented using a Fibonacci heap.[30] Here, $\delta_{\mathcal{T}_{\phi,H}}$ is the total number of transitions in the product transition system. To express this computational complexity in terms of problem data, namely, the number of cells N^{C} and the number of atomic propositions N^{R} in the given LTL$_{-X}$ specification, note first that that $|\mathcal{T}_{\phi,H}| = |V_H||S|$, and $|\delta_{\mathcal{T}_{\phi,H}}| \approx |E_H||S|$. The minimum number of states in the Büchi automaton $\mathcal{B}_\phi$ depends on the structure of ϕ; for specifications of typical relevance in aircraft guidance applications (see Section VI for illustrative examples), we consider $|S| \approx N^{\mathrm{R}} + 1$, as reported in the literature.[31] Next, we note that the number

of edges in the cell decomposition graph $\mathcal{G} = \mathcal{G}_0$ is of the same order as the number of cells, i.e. $E \approx 2N^{\mathrm{C}}$, for the specific case of 4-connected rectangular cells considered in this paper. Next, note that $|V_H| \approx N^{\mathrm{C}}4^H$, and that $E_H \approx 4|V_H| \approx N^{\mathrm{C}}4^{H+1}$. It follows that the worst-case complexity of the proposed algorithm is $\mathcal{O}\left(N^{\mathrm{T}}4^{H+1} + N^{\mathrm{T}}4^H \log\left(N^{\mathrm{T}}4^H\right)\right)$, where $N^{\mathrm{T}} := N^{\mathrm{C}}(N^{\mathrm{R}} + 1)$.

V. Numerical Computation of Edge Transition Costs

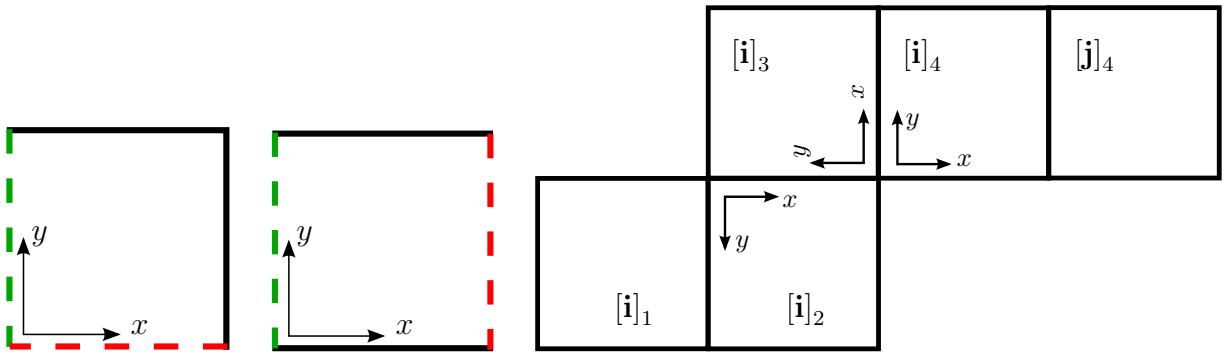
In this section, we address the computation of the sets $\mathcal{R}(\cdot)$, $\mathcal{Q}(\cdot)$, and $\mathcal{S}(\cdot)$ that appear in the edge transition cost definition (7)–(9).

Admissible state trajectories of the vehicle model in Section II are continuously differentiable planar curves with a speed profile, such that the curvature at any point on these curves satisfies the speed-dependent upper bound

$$\kappa_{\max} := \frac{\sqrt{a^2 - \dot{\nu}^2}}{\nu^2 a \rho}. \tag{12}$$

Recall that, due to the control input constraint (2), $|\dot{\nu}| \leqslant a$, which ensures that the previous expression for $\kappa_{\max}$ is real. Based on this observation, and previously reported geometric analysis of curvature-bounded paths,[32] we discuss a method to numerically compute the edge transition costs defined in (7)–(9). In the analysis that follows, we assume that the workspace has been partitioned using square cells of uniform size. Without loss of generality, we assume that the size of each side of any cell is 1 unit.

First, recall that each edge $(\mathbf{i}, \mathbf{j}) \in E_H$ is associated with a sequence of $H+2$ successively adjacent cells, namely, the sequence $\left(\mathsf{cell}([\mathbf{i}]_1), \ldots, \mathsf{cell}([\mathbf{i}]_{H+1}), \mathsf{cell}([\mathbf{j}]_{H+1})\right)$ as illustrated in Fig. 5(c). Each cell in this sequence, called a *tile*,[24] admits either traversal across adjacent faces (e.g. $\mathsf{cell}([\mathbf{i}]_2)$ and $\mathsf{cell}([\mathbf{i}]_3)$ in Fig. 5(c)), or traversal across opposite faces (e.g. $\mathsf{cell}([\mathbf{i}]_4)$ in Fig. 5(c)). We consider canonical forms of these cell traversal types, and attach local, right-handed coordinate axes systems as shown in Figs. 5(a) and 5(b). Similarly, local coordinate axes systems are attached to each cell such that (a) for traversal across adjacent faces, the x-axis coincides with the "exiting" face, and (b) for either type of traversal, the y-axis coincides with the "entering" face, as shown in Fig. 5(c). Note that each local axes system can be transformed via rigid rotations and/or reflections to either of the canonical forms shown in Figs. 5(a) and 5(b).



(a) Adjacent faces. (b) Opposite faces. (c) Illustration of a sequence of cells associated with $(\mathbf{i}, \mathbf{j}) \in E_3$.

Figure 5. Illustration of canonical traversals (adjacent and opposite) and local coordinate axes systems. In (a) and (b), the green- and red colored dotted lines indicate, respectively, the entering and exiting faces of traversal of the cell.

The advantage of attaching these local coordinate axes systems to each cell is that, for $k = 1, \ldots, H$, for every state $\xi_s \in \mathsf{cell}([\mathbf{i}]_k) \cap \mathsf{cell}([\mathbf{i}]_{k+1})$, the local coordinates of $\mathsf{x}(\xi_s)$ are of the form $(0, w)$, where $0 \leqslant w \leqslant 1$. Consequently, the local coordinates of the states in the previously introduced sets $\mathcal{Q}(\cdot)$ and $\mathcal{S}(\cdot)$ reside in the parallelepiped $\mathfrak{S} := [0, 1] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [\nu_{\min}, \nu_{\max}]$. This observation is crucial for developing a concise representation of (approximations of) the forward- and backward reachable sets involved in (7)–(9), and consequently enable fast online computations thereof.

The preprocessing stage of the proposed approach is summarized as follows. First, we generate a library of tiles for each relevant value of $H = 1, 2, \ldots, .$ Next, for each of these tiles in this library, we consider two copies of the parallelepiped $\mathfrak{S} := [0, 1] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [\nu_{\min}, \nu_{\max}]$ associated with, respectively, the second and third cells in the tile. These copies are denoted, respectively, by $\mathfrak{S}_1$ and $\mathfrak{S}_2$. Next, we compute numerical approximations of the sets of the sets $\mathcal{R}(\cdot)$, $\mathcal{Q}(\cdot)$, and $\mathcal{S}(\cdot)$ using geometric analysis of curvature-bounded planar curves. Next, we partition the parallelepiped $\mathfrak{S}$ into uniformly sized regions. Finally, we compute intersections or inclusions of these regions with the sets $\mathcal{R}(\cdot)$, $\mathcal{Q}(\cdot)$, and $\mathcal{S}(\cdot)$, as appropriate, to determine a connectivity matrix of regions within $\mathfrak{S}_1$ to those within $\mathfrak{S}_2$. This connectivity matrix is stored and looked up during online computations of edge transition costs in $\mathcal{G}_H$. In what follows, we provide additional details of each of the steps outlined above.

V-A. Library of Tiles

Recall that for each $H \geqslant 1$, the number of cells in any tileis $H + 2$. It is straightforward to generate a library of tiles for each value of H by enumerating all possible permutations of successive traversal types. During real-time computations, the sequence of cells associated with any edge in $\mathcal{G}_H$ can be uniquely mapped to a tile in this library via rigid transformations. Examples of tile libraries are shown in Appendix A.

V-B. Geometric Computation of Forward- and Backward Reachable Sets

Computations of the backward reachable set $\mathcal{Q}(\cdot)$ can be performed using previously reported analysis[32] of curvature-bounded curves in rectangular channels similar to the regions enclosed by tiles. The details of this analysis are available in the literature,[32,33] and are therefore omitted from this paper. Here, it suffices to note that, for the tile associated with an edge $(\mathbf{i}, \mathbf{j}) \in E_H$ and a prespecified $\kappa > 0$,, the results of this analysis are pointwise numerical values of four piecewise continuous functions $\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2 : [0, 1] \to \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ such that:

1. For every point with coordinates $(0, w)$ on the cell boundary $\mathsf{cell}([\mathbf{i}]_1) \cap \mathsf{cell}([\mathbf{i}]_2)$, there exists a continuously differentiable curve with maximum curvature κ traversing the remainder of the tile if the initial tangent angle of this curve lies in the interval $[\underline{\alpha}_1(w), \overline{\alpha}_1(w)]$. All of the quantities involved here are expressed in the local coordinate axes system attached to the second cell of the tile, i.e. $\mathsf{cell}([\mathbf{i}_2])$.

2. For every point with coordinates $(0, w)$ on the cell boundary $\mathsf{cell}([\mathbf{i}]_2) \cap \mathsf{cell}([\mathbf{i}]_3)$, there exists a continuously differentiable curve with maximum curvature κ traversing the remainder of the tile if the initial tangent angle of this curve lies in the interval $[\underline{\alpha}_2(w), \overline{\alpha}_2(w)]$. All of the quantities involved here are expressed in the local coordinate axes system attached to the third cell of the tile, i.e. $\mathsf{cell}([\mathbf{i}_3])$.

Note that these functions represent the backward reachable sets $\mathcal{Q}(\cdot)$ defined in Section III-A.

The forward reachability sets $\mathcal{R}(\xi_{\mathrm{s}})$ are relatively easier to compute.[34] In the present context, we leverage the assumption (stated in Section II) $\rho\nu_{\min} > 3$ and obtain conservative approximations to these sets in the form of regions described by $[\underline{w}, \overline{w}] \times \left[\underline{\beta}, \overline{\beta}\right] \times [\underline{\nu}, \overline{\nu}]$. The derivation and exact expressions for the quantities $\underline{w}, \overline{w}, \underline{\beta}, \overline{\beta}, \underline{\nu}, \overline{\nu}$, all of which depend

on ξ_s, are lengthy and cumbersome, and are therefore omitted. For the reader's convenience, MATLAB® code implementing these computations is made available at `http://users.wpi.edu/~rvcowlagi/software.html`.

The preprocessing stage in the proposed route-planning algorithm involves the computation of these forward- and backward reachable set for each element of the tile library for each value of H. Furthermore, recall that the curvature constraints of interest for the vehicle model considered in this paper are speed-dependent. Therefore, these computations are in fact performed for several different values of the curvature bound. The details of these preprocessing computations with precise index notations are discussed next.

Note that the results of computations of these forward and backward reachable sets constitute large volumes of numerical data, which is not advisable for storage and online lookup (even if the computations themselves are preprocessed offline). To resolve this issue, we store only an "abstraction" of the relations between these reachable sets in the form of sparse connectivity matrices, as explained next.

V-C. Regions in $\mathfrak{S}$ and Connectivity

Consider an edge $(\mathbf{i}, \mathbf{j}) \in E_H$, and the associated tile. As previously noted, we consider copies $\mathfrak{S}_1$ and $\mathfrak{S}_2$ of the parallelepiped $[0,1] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [\nu_{\min}, \nu_{\max}]$ associated with, respectively, the second and third cells in the tile. Each of $\mathfrak{S}_1$ and $\mathfrak{S}_2$ are partitioned into uniformly sized regions, i.e. each region in this partition is itself a parallelepiped of dimensions $\Delta_w := \frac{1}{N^w}$, $\Delta_\psi := \frac{1}{N^\psi}$, and $\Delta_\nu := \frac{1}{N^\nu}$, where $N^w, N^\psi, N^\nu \in \mathbb{Z}_{\geqslant 0}$ are prespecified. We denote these regions by $\sigma_1[m], \sigma_2[m]$, where $m = 0, 1, \ldots, (N^w N^\psi N^\nu - 1)$, and the subscripts 1 and 2 indicate to which of the parallelepipeds $\mathfrak{S}_1$ and $\mathfrak{S}_2$ the regions belong. Each region $\sigma_n[m]$, $n \in \{1, 2\}$ is defined by the Cartesian product $\left[\underline{\sigma}_n^w[m], \overline{\sigma}_n^w[m]\right] \times \left[\underline{\sigma}_n^\psi[m], \overline{\sigma}_n^\psi[m]\right] \times \left[\underline{\sigma}_n^\nu[m], \overline{\sigma}_n^\nu[m]\right]$, where

$$\underline{\sigma}_n^w[m] := \mathrm{mod}(\mathrm{mod}(m, N^\nu N^\psi), N^\psi)\Delta_w, \qquad \overline{\sigma}_n^w[m] := \underline{\sigma}_n^w[m] + \Delta_w, \tag{13}$$

$$\underline{\sigma}_n^\psi[m] := \left\lfloor \frac{\mathrm{mod}(m, N^w N^\psi)}{N^\psi} \right\rfloor \Delta_\psi, \qquad \overline{\sigma}_n^\psi[m] := \underline{\sigma}_n^\psi[m] + \Delta_\psi, \tag{14}$$

$$\underline{\sigma}_n^\nu[m] := \left\lfloor \frac{m}{N^\psi N^w} \right\rfloor \Delta_\nu, \qquad \overline{\sigma}_n^\nu[m] := \underline{\sigma}_n^\nu[m] + \Delta_\nu. \tag{15}$$

We associate with each tile a sparse square connectivity matrix C with $N^w N^\psi N^\nu$ rows. The preprocessing stage of the proposed route-planning algorithm then consists of the fol-

lowing computations to be performed for each tile for each value of H:

1. Perform curvature-bounded traversal analysis[32] with N^ν different curvature bounds, and for each determine numerical approximations of the four functions discussed in Section V-B. Denote these functions by $\underline{\alpha}_1^\ell, \overline{\alpha}_1^\ell, \underline{\alpha}_2^\ell, \overline{\alpha}_2^\ell$, where $\ell = 0, 1, \ldots, (N^\nu - 1)$. The different curvature bounds to be used for this analysis are the following:

$$\kappa_{\max,\ell} := \left((\overline{\sigma}_n^\nu[m])^2 \rho\right)^{-1}, \quad m = 0, 1, \ldots, (N^w N^\psi N^\nu - 1). \tag{16}$$

   Note that although the index m takes values in a larger range compared to the index ℓ, by (15), the number of unique values of the right hand side of (16) is N^ν. The expression (16) for curvature bounds arises from the previously discussed expression (12). Here, since the traversal analysis is performed for a range of speeds, the dependence on tangential acceleration can be ignored in favor of conservative approximations to the speed as necessary.

2. Determine an index set $\mathbf{m}_2 \subseteq \{0, 1, \ldots, (N^w N^\psi N^\nu - 1)\}$ such that

$$\sigma_2[m] \cap \Big(\bigcup_{w \in [0,1]} \left[\underline{\alpha}_2^\ell(w), \overline{\alpha}_2^\ell(w)\right] \Big) \neq \varnothing, \quad \text{for each } m \in \mathbf{m_2}, \text{ where } \ell := \left\lfloor \frac{m}{N^\psi N^w} \right\rfloor + 1.$$

   Informally, the index set $\mathbf{m}_2$ indicates the regions $\sigma_2[m]$ that have a nonempty intersection with the backward reachable set $\mathcal{Q}(\mathbf{j})$.

3. Determine an index set $\mathbf{m}_1 \subseteq \{0, 1, \ldots, (N^w N^\psi N^\nu - 1)\}$ such that

$$\sigma_1[m] \subseteq \bigcup_{w \in [0,1]} \left[\underline{\alpha}_1^\ell(w), \overline{\alpha}_1^\ell(w)\right], \quad \text{for each } m \in \mathbf{m_1}, \text{ where } \ell := \left\lfloor \frac{m}{N^\psi N^w} \right\rfloor + 1.$$

   Informally, the index set $\mathbf{m}_1$ indicates the regions $\sigma_1[m]$ that are included in the backward reachable set $\mathcal{Q}(\mathbf{i})$.

4. For each $m \in \mathbf{m}_1$,
   (a) Determine $\underline{w}_m, \overline{w}_m, \underline{\beta}_m, \overline{\beta}_m, \underline{\nu}_m, \overline{\nu}_m$, (explicit expressions omitted due to length; MATLAB® code available at `http://users.wpi.edu/~rvcowlagi/software.html`).
   (b) Determine an index set $\mathbf{m}_2^{\mathrm{r}} \subseteq \mathbf{m}_2$ such that

$$\sigma_2[p] \cap \left([\underline{w}_m, \overline{w}_m] \times \left[\underline{\beta}_m, \overline{\beta}_m\right] \times [\underline{\nu}_m, \overline{\nu}_m] \right) \neq \varnothing, \quad \text{for each } p \in \mathbf{m}_2^{\mathrm{r}}. \tag{17}$$

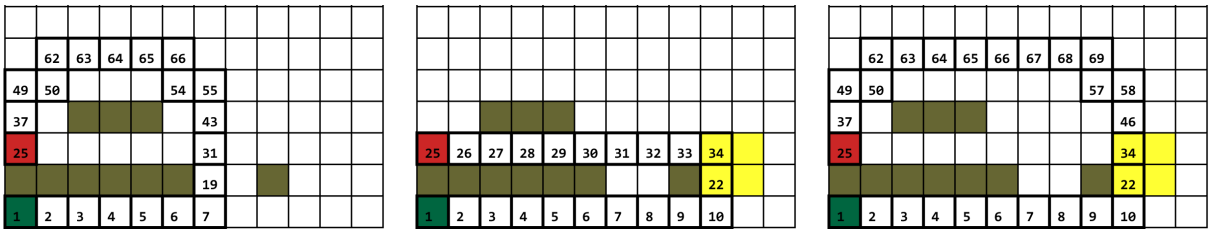
(c) Assign $C_{mp} = 1$, where C_{mp} is the element in the m^{th} row and p^{th} column in C. This step records connectivity between the m^{th} region in $\mathfrak{S}_1$ with the p^{th} region in $\mathfrak{S}_2$.

VI. Illustrative Numerical Simulation Results and Discussion

Figures 6–8 illustrate applications of the proposed route guidance algorithm. In each of these results, a workspace partition with uniformly-sized square cells is considered. For clarity, the indices assigned to all cells are not shown; instead, different colors are used to indicate associations with atomic propositions, as in (4). In what follows, we use the notation of Section II. All of these illustrative examples assume a constant unit speed of traversal, i.e. $u_1 = 0$. For preprocessing the connectivity matrices, the values $N^w = N^\psi = 100$ are used. Also, all of these illustrative examples are set up in a cell decomposition consisting of $N^{\text{C}} = 144$ cells.

For the result shown in Fig. 6(a), the number of atomic propositions is $N^{\text{R}} = 3$. In each of the following examples, the atomic proposition λ_1 is associated with all cells, including those indicated in white color, to establish workspace limits. The cells associated with atomic propositions λ_2, λ_3, and λ_4 are indicated in Fig. 6(a) by gray (obstacles), red, and yellow colors respectively. The dark green cell in the lower left corner is the cell containing $\mathsf{x}(\xi_0)$. The sequence of numbered cells with bold outlines in Figure 6(a) indicate the route obtained as a result of searching the product transition system described in Section IV, for the LTL$_{-X}$ formula $\phi_1 := \Box\lambda_1 \wedge \Box\neg\lambda_2 \wedge \Diamond\lambda_3$. This specification is read *"always λ_1, never λ_2, and eventually λ_3"*. Informally, this specification requires the vehicle to avoid the gray-colored regions and visit the red-colored region, and is equivalent to the standard motion-planning problem of reaching a destination while avoiding obstacles. For the result indicated in Figure 6(a), the control input constraint parameter ρ is set to $\rho = 3$ units. Here, the unit of measurement is the size of a cell. The numbers indicated within cells in the resultant route are the indices assigned to those cells.

The sequences of numbered cells with bold outlines in Figs. 6(b) and 6(c) indicate the resultant route for satisfying the LTL$_{-X}$ specification $\phi_2 := \phi_1 \wedge \Diamond\lambda_4$. Informally, this specification requires the vehicle to avoid the gray-colored regions, and visit the red- and



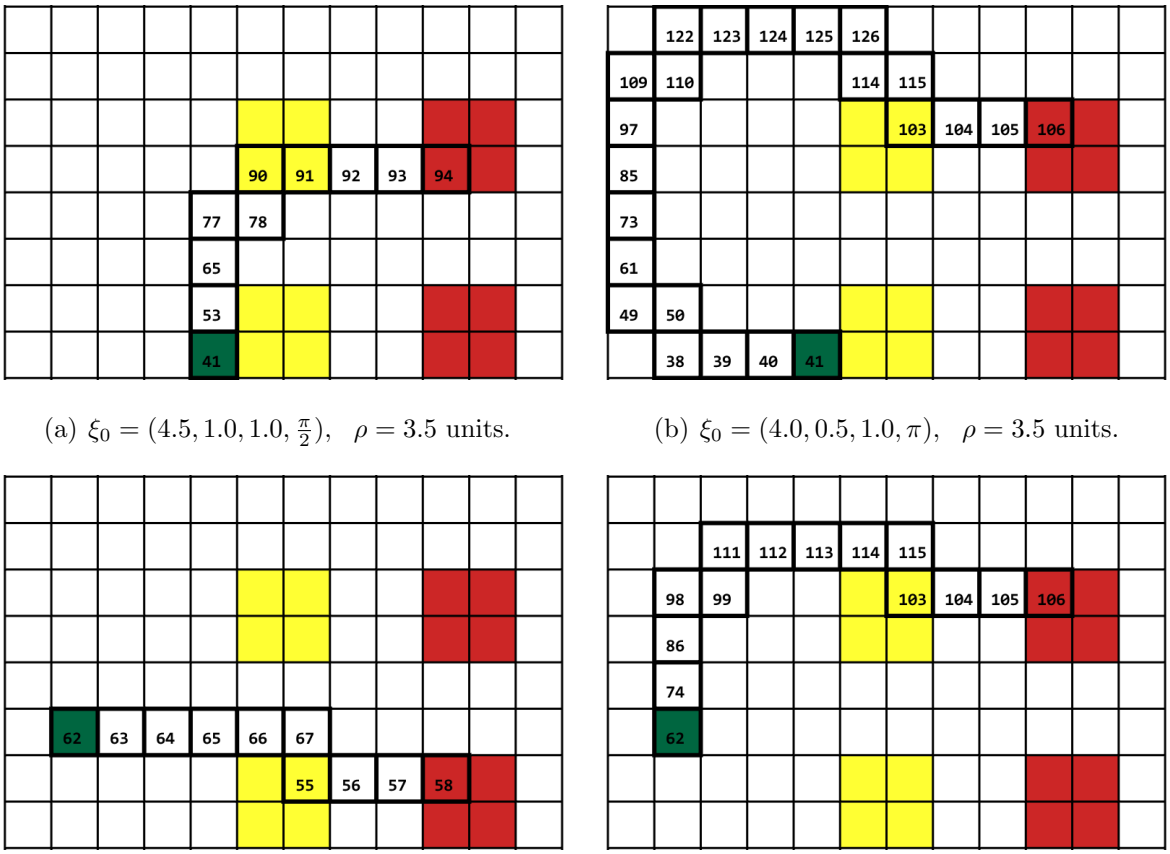
(a) Spec. ϕ_1 and $\rho = 3.0$ units. (b) Spec. ϕ_2 and $\rho \leqslant \frac{1}{1.55}$ units. (c) Spec. ϕ_2 and $\rho = 3.0$ units.

Figure 6. Application of the proposed approach: illustration of the effects on the resultant path of changes in the control input constraint. The numerical values ρ are in dimensionless distance units, where 1 unit is equal to the side of a cell in the uniform decomposition.

yellow-colored regions both. The different results in Figs. 6(b) and 6(c) are a consequence of changing the parameter ρ. In particular, for the result indicated in Fig. 6(b), $\rho \leqslant \frac{1}{1.55}$ units, whereas for the result indicated in Fig. 6(c) $\rho = 3$ units. Notice that the *same* LTL_{-X} specification is satisfied by two markedly different paths due to the different input constraints. The result shown in Fig. 6(b) is computed by ignoring the vehicle model altogether during route-planning and relies on the previously known result[35] that if a channel is sufficiently "wide," (specifically, at least 1.55 times the minimum radius of turn) then a curvature-bounded curve is guaranteed to traverse it from any initial condition.

For the result shown in Fig. 7, the number of atomic propositions is $N^{\text{R}} = 3$. The cells associated with atomic propositions λ_2 and λ_3 are indicated in Fig. 7 by red and yellow colors respectively. Informally, this specification requires the vehicle visit the red- and yellow-colored regions both. However, the order of visit is not specified, and there is no preference for *which* of the two yellow- or two red-colored regions is to be visited. The dark green cell near the center is the cell containing $\mathsf{x}(\xi_0)$. The sequence of numbered cells with bold outlines in each of Figs. 7(a)–7(d) indicate the route obtained as a result of searching the product transition system, for the LTL_{-X} formula $\phi_3 := \Box\lambda_1 \wedge \Diamond\lambda_2 \wedge \Diamond\lambda_3$. Notice that the regions associated with the propositions λ_2 and λ_3 (the red- and yellow-colored regions, respectively) are unions of disconnected subregions. The different resultant paths in each of Figs. 7(a)–7(d) are due to different initial states, as indicated in the captions.

Two additional results are shown in Fig. 8 to illustrate the impact of nonholonomic constraints on the manner in which the given LTL specifications are satisfied, and of the



(a) $\xi_0 = (4.5, 1.0, 1.0, \frac{\pi}{2})$, $\rho = 3.5$ units.

(b) $\xi_0 = (4.0, 0.5, 1.0, \pi)$, $\rho = 3.5$ units.

(c) $\xi_0 = (2.0, 2.5, 1.0, 0)$, $\rho = 3.5$ units.

(d) $\xi_0 = (1.5, 3.0, 1.0, \frac{\pi}{2})$, $\rho = 3.5$ units.

Figure 7. Application of the proposed approach: illustration of the effects on the resultant path of changes in the initial state. Here, the LTL$_{-X}$ specification is ϕ_3 (described in text). The numerical values of position coordinates and of ρ are in dimensionless distance units, where 1 unit is equal to the side of a cell in the uniform decomposition.

effectiveness of the proposed route-planning algorithm in handling these situations. For the result shown in Fig. 8(a), the number of atomic propositions is $N^{\mathrm{R}} = 4$, with atomic proposition λ_2 associated with gray regions, and propositions λ_3, λ_4 associated with the red- and yellow-colored regions respectively. The dark green-colored cell at the bottom left is the cell containing $\mathsf{x}(\xi_0)$. The sequence of numbered cells with bold outlines in Fig. 8(a) indicates the route resulting from the proposed algorithm for satisfying the LTL$_{-X}$ specification $\phi_4 := \phi_2 \wedge (\neg\lambda_4 \triangleright \lambda_3)$, which requires not only that the regions associated with λ_3 and λ_4 be both visited, but also that the region associated with λ_3 be visited first. The result shown in Fig. 8 is of significance for intelligence, surveillance, and reconnaissance UAV operations, where mission requirements may dictate the order of visit to different regions.

For the result shown in Fig. 8(b), the number of atomic propositions is $N^{\mathrm{R}} = 5$, with atomic proposition λ_2, associated with gray regions, the propositions λ_3 and λ_4 associated with the red- and yellow-colored regions, respectively, and the proposition associated with the two green-colored regions. The green-colored cell at the bottom left is the cell containing $\mathsf{x}(\xi_0)$. The LTL_{-X} specification $\phi_5 := \phi_4 \wedge \Diamond\Box\lambda_5$ is considered.

Informally, this specification requires that the gray-colored regions be avoided, that the red-colored region be visited, followed by the yellow-colored region, and that the route terminate in either one of the green-colored regions. This specification is simple, yet of immense practical significance: the sub-formula $(\Diamond\Box\lambda_5)$ codifies the common – and necessary – mission requirement of *"return UAV to base,"* where the "base" location need not be unique (e.g. there may be multiple landing strips available).

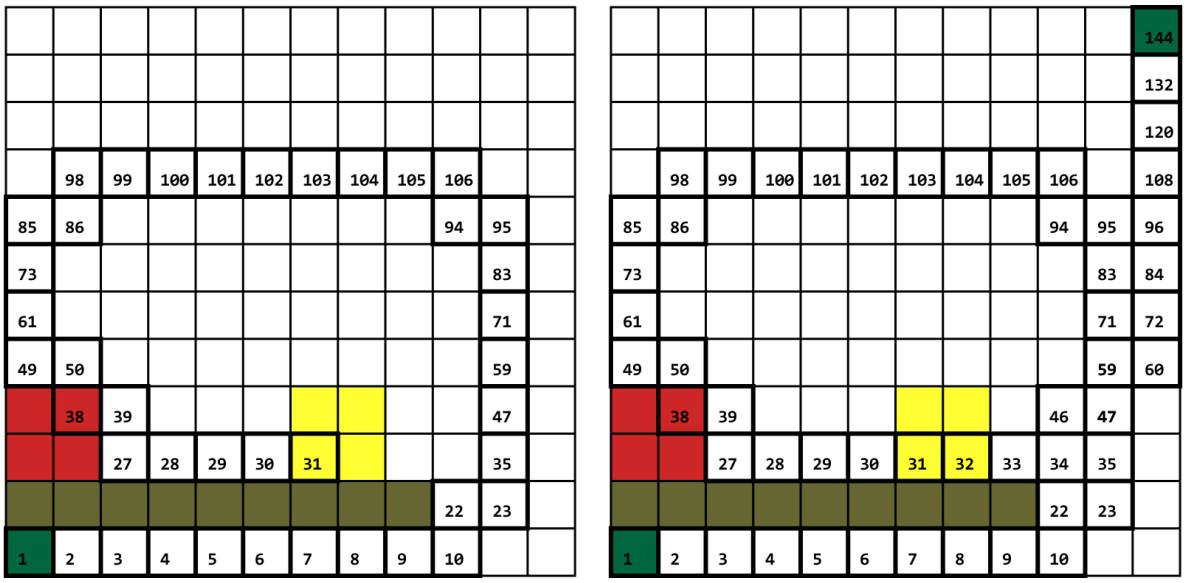
The sequence of numbered cells with bold outlines indicates the route determined by the proposed route-planning algorithm. This route includes a cycle, and is defined by the sequence of cells

$$(1, \ldots, 10, 22, 23, 35, \ldots, 95, 94, \ldots, 33, 34, 46, 47, 59, 60, 72 \ldots, 144).$$

Notice that this result is significantly impacted by the control input constraints. Specifically, the route for a holonomic vehicle in this case may have simply involved returning to the green-colored cell (indicated with index 1) in the bottom left after visiting the red- and yellow-colored regions. However, such a route is infeasible with the control input constraint indicated in the figure caption. Furthermore, such a planned route (possibly computed by a naive route-planning algorithm) would have been a particularly adverse plan, considering the UAV's kinematic constraints. This example therefore indicates the benefit offered by the proposed approach in finding routes that not only satisfy given LTL specifications, but are also compatible with the vehicle model and its constraints.

Figures 6–8 represent the results of executing the proposed route-planning algorithm with parameter $H = 5$. The algorithm is implemented in the MATLAB® (version R2014b), and executed on a desktop computer with an Intel® Core™ i7 2.80 GHz processor, 16 GB RAM, and the Windows® 7 Enterprise 64-bit operating system. Representative computation times involved in the proposed approach are provided in Tables 1 and 2.

The numbers of vertices in lifted graphs $\mathcal{G}_H$, for several different values of the parameter



(a) Spec ϕ_4 and $\rho = 3.5$ units.

(b) Spec ϕ_5 and $\rho = 3.5$ units.

Figure 8. Additional examples that illustrate the significant impact of nonholonomic motion constraints on the manner in which the given LTL specifications are satisfied (details in text).

H, are provided in Table 1. The computation time τ_{tplg} in the third row of Table 1 is the time required to record the topology of $\mathcal{G}_H$, i.e. to find and store in memory a list of the vertices and edges in the lifted graph. This may be considered a part of the offline preprocessing stage, because the edge transition costs in $\mathcal{G}_H$ are not assigned at this stage.

The computation times for setting up and executing the proposed route-planning algorithm for the examples shown in Figs. 6–8 are provided in Table 2. Here, τ_{setup} is the total time required to record the lifted graph topology, to construct the Büchi automaton (using the LTL2BA algorithm,[12] and its MATLAB® adaptation[36]), and to enlist and store transitions in the product transition system $\mathcal{T}_{\phi,H}$. The time τ^0_{search} is the time required to search the product transition system using Dijkstra's algorithm to find a route. The times τ^1_{search} and τ^2_{search} are the times required to search the product transition system using an A*-like informed search algorithm with a search heuristic. A full discussion of these search heuristics is beyond the scope of this paper, and the τ^1_{search} and τ^2_{search} are provided to indicate potential improvements in the execution time of the proposed algorithm. Briefly, these search heuristics are based on a preliminary search in the product transition system $\mathcal{T}_{\phi,0}$, which executes quickly. The data reported in Tables 1 and 2 are averages over three simulation trials for

each specification, each value of the parameter H, and each value of the control constraint ρ for specifications ϕ_1 and ϕ_2.

H	0	1	2	3	4	5	6	7
$\lvert V_H \rvert$	144	528	1,448	3,072	6,832	15,032	33,088	71,200
τ_{tplg} (s)	—	0.0238	0.0895	0.2633	0.9265	3.347	12.95	53.46

Table 1. Number of vertices in the lifted graph, for various values of the parameter H, and the time required to record the graph topology.

VI-A. Discussion

The proposed route-planning algorithm offers several advantages compared to the state-of-the-art. First, a significant portion of the computations involved can be preprocessed (as discussed in Section V), which reduces the online computational burden. For the numerical simulation results shown in Figures 6–8, the time to execute the preprocessing step with $N^w = N^\psi = 100, N^\nu = 1$, $H = 1, \ldots, 7$, and $\rho = 3.5$, on the aforementioned desktop computer, is approximately 65 minutes. Connectivity matrices for each of a total of 254 tiles (unique up to rigid transformations) are stored. The total number of non-zero elements in all of these matrices is approximately 64.21×10^6, which requires memory space of approximately 32 MB using the sparse matrix data structure provided by MATLAB®.

Whereas the execution times reported in Table 2 are of the order of seconds or tens of seconds, we claim that significant portions of these execution times are merely due to the computational overhead introduced by MATLAB®. This claim is corroborated by the fact that the sizes of the product transition systems, in terms of numbers of states, indicated in Table 2 are relatively small in comparison to the typical graph sizes (of the order of 10^7 vertices) that can be searched by high-performance software libraries within a few tens of seconds.[37,38] The main difficulty in the proposed approach is that the transition costs in the product transition system depend on computationally intensive reachability calculations. However, the offline preprocessing stage as discussed in Section V enables the determination of these transition costs via a simple lookup table. Therefore, it is possible to reduce these execution times by several orders of magnitude by appropriately implementing the

Specification →	ϕ_1			ϕ_2			ϕ_3	ϕ_4	ϕ_5
$\rho \rightarrow$	3.0	4.0	5.0	3.0	4.0	5.0	3.5	3.5	3.5
H = 3, $\lvert\mathcal{T}_{\phi,H}\rvert$	6,146			12,292			12,292	9,219	18,438
$\lvert\delta_\phi H\rvert$	11,110			22,892			32,308	17,771	17,963
τ_{setup} (s)	2.429			5.040			4.917	—	—
τ^0_{search} (s)	6.391	3.705	—	18.82	6.992	2.258	0.9605	—	—
τ^1_{search} (s)	1.049	2.036	—	11.47	6.076	2.251	0.6679	—	—
H = 4, $\lvert\mathcal{T}_{\phi,H}\rvert$	13,666			27,332			27,332	20,499	40,998
$\lvert\delta_\phi H\rvert$	23,799			49,114			73,608	38,496	38,904
τ_{setup} (s)	6.545			13.73			13.73	—	—
τ^0_{search} (s)	14.88	6.083	1.985	42.09	13.95	3.814	1.398	—	—
τ^1_{search} (s)	2.272	2.758	1.995	24.88	10.02	3.563	0.8846	—	—
H = 5, $\lvert\mathcal{T}_{\phi,H}\rvert$	30,066			60,132			60,132	45,099	90,198
$\lvert\delta_\phi H\rvert$	51,114			105,660			167,582	83,563	84,475
τ_{setup} (s)	19.84			39.12			37.64	35.68	71.56
τ^0_{search} (s)	22.42	12.90	6.205	69.69	34.29	9.443	1.436	38.08	66.78
τ^1_{search} (s)	2.421	3.462	6.028	37.20	21.84	9.132	0.9781	35.90	84.27
τ^2_{search} (s)	Experiment not performed.							31.65	49.26

Table 2. Execution times for the numerical simulation examples discussed in Section VI. The blank entries for τ^0_{search} and τ^1_{search} for $H = 3, 4$ indicate that the search returned failure after a finite number of iterations, and no route was found.

proposed algorithm with a low-level programming language such as C++, and by utilizing high-performance open-source software libraries. Appropriate search heuristics can also significantly reduce the execution times, and enable real-time implementations of the proposed algorithm.

Second, as previously noted for the example shown in Fig. 7, the proposed approach allows for the association of disconnected regions in the workspace with a single atomic proposition. Consequently, the total number of propositions can be reduced, thereby reducing the size of the product transition system. This ability is not available in other approaches reported in the literature, e.g.,[17,19] which depend on partitioning the vehicle's *state space* instead of its workspace (output space).

Third, the proposed approach does not use up control authority for the sake of generating a discrete abstraction of the vehicle dynamical model, as is often done in the literature.[4,6,17,19,39] This approach leaves room to accommodate any independent trajectory optimization algorithm to be employed for generating reference trajectories for tracking the planned route. The trajectory optimization problem is subjected to the constraint that the workspace projection of the resulting must remain within the channel associated with the planned route (namely, the channels indicated by numbered cells with bold outlines in Figs. 6–8).

VII. Conclusions and Future Work

In this paper, we discussed a new technique for aircraft route guidance subject to LTL_{-X} specifications. The proposed technique relies on workspace partitioning, which involves partitioning in a space of smaller dimension than the *state* space, as is often done in the literature. The proposed technique relied on the so-called lifted graph, and on apropriate assignments of edge transition costs in the lifted graph. The relationship of these edge transition costs with certain forward- and backward reachable sets of the vehicle model was discussed, and the offline preprocessing of the computations of these sets was emphasized.

The proposed approach is applicable more generally to differentially flat nonlinear systems, especially when control input constraints can be mapped to constraints on the flat output-space trajectories. The control constraints for the vehicle model considered in this

paper were mapped to curvature constraints on the admissible workspace (output space) trajectories. Therefore, analysis of forward- and backward reachable sets with curvature-bounded curves was directly applied for finding lifted graph edge transition costs for this vehicle model. Future work includes the application of the proposed approach for control with temporal logic specifications of other differentially flat nonlinear systems.

Acknowledgments: This research was funded in part by US Air Force SBIR contract # FA8501-14-P-0034 awarded to Aurora Flight Sciences (AFS) Corp., Cambridge, MA, in collaboration with WPI.

References

[1]Pnueli, A., "The Temporal Logic of Programs," *18th Annual Symposium on Foundations of Computer Science*, Providence, RI, USA, October 31 - November 2 1977, pp. 46–57.

[2]Alagar, V. S. and Periasamy, K., *Specification of Software Systems*, Springer-Verlag, London, UK, 2nd ed., 2011.

[3]Baier, C. and Katoen, J.-P., *Principles of Model Checking*, The MIT Press, Cambridge, MA, USA, 2008.

[4]Belta, C., Isler, V., and Pappas, G. J., "Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments," *IEEE Transactions on Robotics*, Vol. 21, No. 5, October 2005, pp. 864 – 874.

[5]Fainekos, G. E., Kress-Gazit, H., and Pappas, G. J., "Hybrid Controllers for Path Planning: A Temporal Logic Approach," *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Spain, 12 – 15 Dec. 2005, pp. 4885 – 4890.

[6]Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G. J., "Symbolic Planning and Control of Robot Motion," *IEEE Robotics and Automation Magazine*, March 2007, pp. 61 – 70.

[7]Alur, R., Henzinger, T. A., Lafferriere, G., and Pappas, G. J., "Discrete Abstractions of Hybrid Systems," *Proceedings of the IEEE*, Vol. 88, No. 7, July 2000, pp. 971–984.

[8]Tabuada, P., *Verification and Control of Hybrid Systems: A Symbolic Approach*, Springer, 2008.

[9]Wolper, P., Vardi, M., and Sistla, A., "Resoning about Infinite Computations," *Proceedings of the 24th Symposium on Foundations of Computer Science*, Tucson, AZ, 1983, pp. 185–194.

[10]Wolper, P., "Constructing automata from temporal logic formulas: A tutorial," *Formal Methods Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science*, Springer-Verlag, New York, NY, 2001.

[11]Holzmann, G., "The model checker SPIN," *IEEE Transactions on Software Engineering*, Vol. 23, No. 5, 1997, pp. 279–295.

[12]Gastin, P. and Oddoux, D., "Fast LTL to Büchi automata Translation," *Proceedings of the 13th Conference on Computer Aided Verification (CAV' 01)*, edited by H. C. G. Berry and A. Finkel, Vol. 2102 of *Lecture Notes in Computer Science*, Springer-Verlag, New York, 2001, pp. 53–65.

[13]Ulusoy, A., Smith, S. L., Ding, X. C., Belta, C., and Rus, D., "Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints," *International Journal of Robotics Research*, Vol. 32, No. 8, 2013, pp. 889–911.

[14]Ding, X. C., Lazar, M., and Belta, C., "LTL Receding Horizon Control for Finite Deterministic Systems," *Automatica*, Vol. 50, 2014, pp. 399–408.

[15]DeCastro, J. A. and Kress-Gazit, H., "Synthesis of nonlinear continuous controllers for verifiably correct high-level, reactive behaviors," *International Journal of Robotics Research*, Vol. 34, No. 3, 2015, pp. 378–394.

[16]Kloetzer, M. and Belta, C., "Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions," *IEEE Transaction on Robotics*, Vol. 23, No. 2, 2007, pp. 320–330.

[17]Kloetzer, M. and Belta, C., "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications," *IEEE Transactions on Automatic Control*, Vol. 53, No. 1, February 2008, pp. 287–297.

[18]Yordanov, Y., Tůmova, J., Černá, I., Barnat, J., and Belta, C., "Temporal Logic Control of Discrete-Time Piecewise Affine Systems," *IEEE Transactions on Automatic Control*, Vol. 57, No. 6, 2012, pp. 1491–1505.

[19]Zamani, M., Pola, G., Mazo, M., and Tabuada, P., "Symbolic Models for Nonlinear Control Systems Without Stability Assumptions," *IEEE Transactions on Automatic Control*, Vol. 57, No. 7, 2012, pp. 1804–1809.

[20]Wolff, E. M., Topcu, U., and Murray, R. M., "Optimization-based Control of Nonlinear Systems with Linear Temporal Logic Specifications," *Proceedings of the 2014 International Conference on Robotics and Automation*, Hong Kong, China, May 31 – June 7 2014, pp. 5319–5325.

[21]Kress-Gazit, H., Conner, D. C., Choset, H., Rizzi, A. A., and Pappas, G. J., "Courteous Cars: Decentralized Multiagent Traffic Coordination," *IEEE Robotics & Automation Magazine*, Vol. 15, 2008, pp. 30–38.

[22]Coogan, S. and Arcak, M., "Efficient finite abstraction of mixed monotone systems," *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, Seattle, WA, USA, April 14 – 16 2015, pp. 58–67.

[23]Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–204.

[24]Cowlagi, R. V. and Tsiotras, P., "Hierarchical Motion Planning with Dynamical Feasibility Guarantees for Mobile Robotic Vehicles," *IEEE Transactions on Robotics*, Vol. 28, No. 2, 2012, pp. 379 – 395.

[25]Tabuada, P. and Pappas, G. J., "Model Checking LTL over Controllable Linear Systems is Decidable," *Hybrid Systems: Computation & Control*, edited by O. Maler and A. Pnueli, LNCS 2623, Springer-Verlag, Berlin, 2003, pp. 498–513.

[26]Zhang, Z. and Cowlagi, R. V., "Incremental Path Repair in Hierarchical Motion-Planning with Dynamic Feasibility Guarantees for Mobile Robotic Vehicles," *European Control Conference ECC'15*, Linz, Austria, July 15 – 17 2015.

[27]Bertsekas, D. P. and Rhodes, I. B., "On the Minimax Reachability of Target Sets and Target Tubes," *Automatica*, Vol. 7, 1971, pp. 233–247.

[28]Hopcroft, J. E., Motwani, R., and Ullman, J. D., *Introduction to automata theory, languages, and computation*, Addison-Wesley, Boston, MA, USA, 2nd ed., 2001.

[29]Bertsekas, D. P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 2000.

[30]Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, MIT Press, 2nd ed., 2001.

[31]Cichon, J., Czubak, A., and Jasinski, A., "Minimal Büchi automata for certain classes of LTL formulas," *Proceedings of the Fourth International Conference on Dependability of Computer Systems (DepCos-RELCOMEX'09)*, 2009, pp. 17–24.

[32]Cowlagi, R. V. and Tsiotras, P., "Curvature-Bounded Traversability Analysis for Motion Planning of Mobile Robots," *IEEE Transactions on Robotics*, Vol. 30, No. 4, 2014, pp. 1011–1019.

[33]Cowlagi, R. V., *Hierarchical Motion Planning for Autonomous Aerial and Terrestrial Vehicles*, Ph.D. thesis, Georgia Institute of Technology, 2011.

[34]Cockayne, E. J. and Hall, G. W. C., "Plane Motion of a Particle Subject to Curvature Constraints," *SIAM Journal on Control*, Vol. 13, No. 1, 1975, pp. 197–220.

[35]Bereg, S. and Kirkpatrick, D., "Curvature-bounded Traversals of Narrow Corridors," *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, Pisa, Italy, 2005, pp. 278–287.

[36]Belta, C., "LTLCon: Control of linear systems from LTL formulas over linear predicates," Software available online at `http://sites.bu.edu/hyness/software/`.

[37]Dawes, B., Abrahams, D., and Rivers, R., "Boost C++ Libraries," Available online at `http://www.boost.org/`.

[38]Edmonds, N., Breuer, A., Gregor, D., and Lumsdaine, A., "Single-Source Shortest Paths with the Parallel Boost Graph Library," *The Ninth DIMACS Implementation Challenge: The Shortest Path Problem*, Piscataway, NJ, November 2006.

[39]Lindemann, S. R., Hussein, I. I., and LaValle, S. M., "Real Time Feedback Control for Non-holonomic Mobile Robots with Obstacles," *Proceedings of the 45^{th} IEEE Conference on Decision and Control*, San Diego, CA, USA, December 2006, pp. 2406–2411.

Appendix A. Proofs and Additional Technical Details

Proof of Proposition 1. Define $\mathbf{i}_k := (v_k, \ldots, v_{k+H})$, for each $k \in \{0, \ldots, P-H\}$. First, suppose that $\mathcal{J}_H(\mathbf{v}) < \chi$. By Eqns. (8) and (9), it follows that, for each $k \in \{0, \ldots, P-H\}$, $g_H(\mathbf{i}_k, \mathbf{i}_{k+1}) = 1$, and that $\mathcal{S}(\mathbf{i}_{k+1})$ is nonempty. By Eqn. (7), for every state $\xi_s \in \mathcal{S}(\mathbf{i}_{k+1})$, there exists $pre(\xi_s) \in \hat{\mathcal{S}}(\mathbf{i}_k, \mathbf{i}_{k+1}) \subseteq \mathcal{S}(\mathbf{i}_k)$ such that $\xi_s \in \mathcal{R}_{\mathbf{i}_k}(pre(\xi_s))$.

In particular, $\mathcal{S}(\mathbf{i}_{P-H})$ is nonempty, and, by Eqn. (7), $\mathcal{Q}(\mathbf{i}_{P-H})$ is nonempty. By definition, for any state $\xi_{P-H} \in \mathcal{Q}(\mathbf{i}_{P-H})$, there exist $t_{P-H} \in \mathbb{R}_+$ and a control input $u_{P-H} \in \mathcal{U}_{t_{P-H}}$ such that

$$\mathsf{x}(\xi(t; \xi_{P-H}, u_{P-H})) \in \cup_{\ell=1}^{H+1} \mathsf{cell}([I_{P-H}]_\ell) \tag{18}$$

Then we iteratively define $\xi_k := pre(\xi_{k+1})$ for each $k = P-H, \ldots, 0$. Note that $\xi_{k+1} \in \mathcal{R}_{I_k}(\xi_k)$. By definition in Eqn. (6), there exist $t_k \in \mathbb{R}_+$ and $u_k \in \mathcal{U}_{t_k}$ such that $\xi(t; \xi_k, u_k) = \xi_{k+1}$, and

$$\mathsf{x}(\xi(t; \xi_k, u_k)) \in \mathsf{cell}([\mathbf{i}_k]_2), \qquad t \in [0, t_k]. \tag{19}$$

Note that $\xi_k \in \mathcal{S}(\mathbf{i}_k)$, which implies that $\xi_0 \in \mathcal{S}(\mathbf{i}_0) = \xi_0$. Now define the concatenated trajectory

$$\xi^*(t) := \xi(t; \xi_k, u_k), \qquad t \in [\tau_k, \tau_{k+1}], \tag{20}$$

where $\tau_0 = 0$ and $\tau_{k+1} := \tau_k + t_k$, for each $k = 0, \ldots, P-H$. By Eqns. (18) and (19), it follows that $\mathbf{v} = tr(\xi^*, \mathcal{G})$, and, by consequence, that $\mathbf{v} \in \mathcal{L}_\Gamma(\xi_0)$.

To prove the converse, suppose that $\mathbf{v} \in \mathcal{L}_\Gamma(\xi_0)$. By definition, $\mathsf{x}(\xi(0; \xi_0, u)) \in \mathsf{cell}(v_0)$ and $\mathsf{x}(\xi(t_f; \xi_0, u)) \in \mathsf{cell}(v_P)$. Define for each $k \in \{0, \ldots, P-H\}$,

$$\tau_k := \max_{\tau_k \in [0, t_f]} \{\tau_k : \mathsf{x}(\xi(\tau_k; \xi_0, u)) \in \mathsf{cell}([\mathbf{i}_k]_1) \cap \mathsf{cell}([\mathbf{i}_k]_2)\},$$
$$\xi_k^s := \xi(\tau_k; \xi_0, u), \qquad u_k := u|_{[\tau_k, \tau_{k+1}]}.$$

The existence of τ_k is guaranteed by continuity of ξ and by (3). It is easy to see that $\tau_0 = 0$, $\xi_k^s = \xi_0$, and that for each $k \in \{1, \ldots, P-H\}$, $\xi_{k+1}^s \in \mathcal{R}_{\mathbf{i}_k}(\xi_k^s)$. By definition of the sets $\mathcal{Q}(\cdot)$, it is also clear that $\xi_{k+1}^s \in \mathcal{Q}(\mathbf{i}_{k+1})$. Therefore, $\xi_{k+1}^s \in \mathcal{S}(k+1)$, and by Eqns. (7) and (8), $g_H(\mathbf{i}_k) = 1$ for each $k \in \{1, \ldots, P-H\}$. It follows that $\mathcal{J}_H(\mathbf{v}) < \chi$. □

Proof of Theorem 1. By Prop. 1,

$$b_H^0(\Theta_{\mathrm{p}}|_{V_H}) \in \mathcal{L}_\Gamma(\xi_0), \qquad\qquad b_H^0(\Theta_{\mathrm{s}}|_{V_H}) \in \mathcal{L}_\Gamma(\xi_0),$$

which implies that $b_H^0((\Theta_{\mathrm{p}}, \Theta_{\mathrm{s}}, \Theta_{\mathrm{s}}, \ldots)|_{V_H}) \in \mathcal{L}_\Gamma(\xi_0)$. By (5), (10), and (11), the path $\mathbf{v} := b_H^0(\Theta|_{V_H}) \in \mathcal{L}_\mathcal{G}$ defines the word $\overline{\omega}(\mathbf{v}) = (\omega_0, \omega_1, \ldots)$. By definition of $\mathcal{T}_{\phi,H}$, $(\theta_k, \omega_k, \theta_{k+1}) \in \delta_{\mathcal{T}_{\phi,H}}$, and $(\theta_k|_S, \omega_k, \theta_{k+1}|_S) \in \delta_{\mathcal{B}_\phi}$ for each $k \in \mathbb{N}$. Therefore, the word $\omega(\mathbf{v})$ is accepted by the Büchi automaton $\mathcal{B}_\phi$, which in turn means that the path π satisfies the formula ϕ and, by consequence, $\mathbf{v} = b_H^0(\Theta_{\mathrm{p}}|_{V_H}) \in \mathcal{L}_{\Gamma\Phi}$.

To prove the converse, consider a path $\mathbf{v} = (j_0, j_1, \ldots) \in \mathcal{L}_{\Gamma\phi} \subseteq \mathcal{L}_\Gamma \subseteq \mathcal{L}_\mathcal{G}$ such that the length of $\mathbf{v}$ is greater than $H+1$ and there are no cycles of length less than or equal to $H+1$. Because the path satisfies the formula ϕ, the word $\omega(\mathbf{v}) = (\omega_0, \omega_1, \ldots)$, where ω_k is defined in Eqn. (5), is accepted by the Büchi automaton $\mathcal{B}_\phi$. Let $s_0, s_1, \ldots$ be the states of $\mathcal{B}_\phi$ visited in the run associated with the input word $\overline{\omega}(\mathbf{v})$. Also, let $\mathbf{i}_k := (v_k, \ldots, v_{k+1}) \in V_H$, and define $\theta_k := (s_k, \mathbf{i}_k)$ for each $k \in \mathbb{N}$. Clearly, $(\theta_k, \omega_k, \theta_{k+1}) \in \delta_{\mathcal{T}_{\phi,H}}$, and it follows that $\Theta := (\theta_0, \theta_1, \ldots)$ is a run of $\mathcal{T}_{\phi,H}$, and that $\Theta|_{V_H} = b_0^H(\mathbf{v})$, which implies $b_H^0(\Theta|_{V_H}) = \mathbf{v}$. □

Illustration of Tile Library: The libraries of all tiles (unique up to rigid transformations) for $H = 2$ and $H = 3$ are shown in Figs. 9 and 10 respectively.

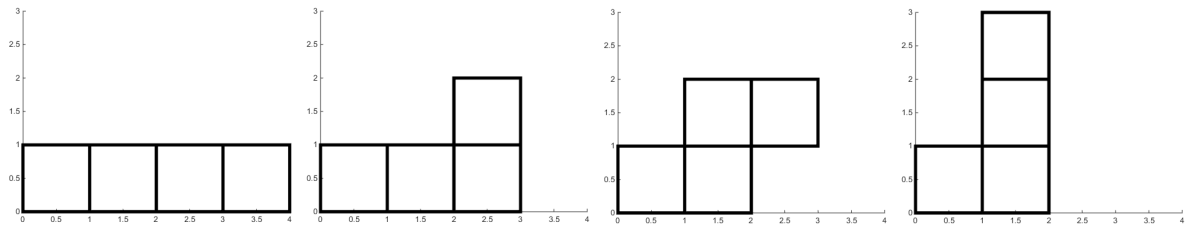


Figure 9. Tile library for $H = 2$, auto-generated using a MATLAB® script.

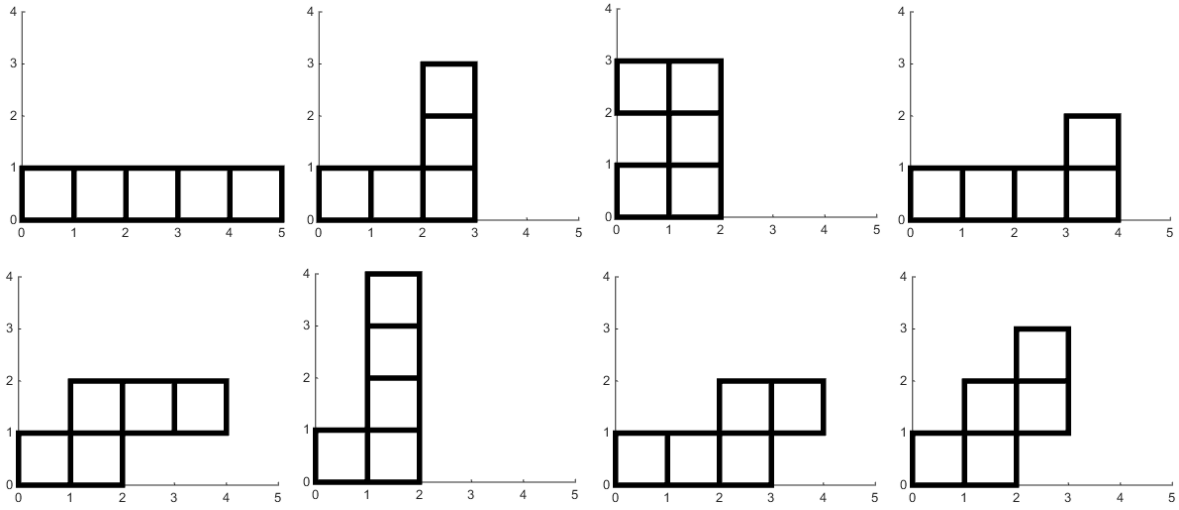


Figure 10. Tile library for $H = 3$, auto-generated using a MATLAB® script.