2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Minimum-violation scLTL motion planning for mobility-on-demand

Cristian-Ioan Vasile[1], Jana Tumova[2], Sertac Karaman[1], Calin Belta[3], Daniela Rus[1]

***Abstract*— This work focuses on integrated routing and motion planning for an autonomous vehicle in a road network. We consider a problem in which customer demands need to be met within desired deadlines, and the rules of the road need to be satisfied. The vehicle might not, however, be able to satisfy these two goals at the same time. We propose a systematic way to compromise between delaying the satisfaction of the given demand and violating the road rules. We utilize scLTL formulas to specify desired behavior and develop a receding horizon approach including a periodically interacting routing algorithm and a RRT*-based motion planner. The proposed solution yields a provably minimum-violation trajectory. An illustrative case study is included.**

I. Introduction

In this paper we develop algorithms that enable a self-driving vehicle to provide a ride to a customer within a certain deadline and while following the traffic rules of the road. These two goals often are not compatible. Speed limits, construction zones, or traffic lights may slow the autonomous vehicle down and prevent it from servicing the demand in time. Some of the traffic rules have to be enforced at all times. However, there are special situations when traffic rules can be viewed with flexibility. For example, one can use the incoming lane to pass a stationary vehicle. Another example is for the case of road work, regular traffic rules may change temporarily to accommodate the changes required by the working crew – for example an illegal U-turn may become legal. Our goal is to compute a plan for the vehicle that is guaranteed to meet the customer deadline within the rules of the road or, if this is not possible, within least violating behavior.

In prior work, we explored similar problems of motion planning under conflicting road rules in [1], [2], where we proposed to formalize the level of road rules violation as a function of their importance and designed a minimum-violation motion planning algorithm. Therein, the customer demand was simple, to travel from A to B, and vehicle routing was not addressed. On the other hand, in [3], we focused solely on vehicle routing in a discretized model of the road network and minimized delays in servicing the customer demands that could not all be serviced within the desired deadlines. In this work, we develop a synergy between motion planning and routing that systematically resolves conflicts between satisfying a set of road rules and servicing a customer demand on time.

We capture the vehicle in the road network using a hierarchical model capturing the dynamics of the vehicle, the road segment it currently traverses, and a high-level discrete abstraction of the road network. The customer demand is a complex task given as a formula in the syntactically co-safe fragment of Linear Temporal Logic (scLTL), and associated with a deadline. The choice of scLTL is motivated by its resemblance to natural language, rigorousness, and expressiveness allowing to formalize a variety of reachability and sequencing tasks, such as "Pick me up at work, then go to the school to pick up the kids and then bring us home. Somewhere on our way, stop by at a shopping mall or a bakery." Rules of the road are expressed in scLTL as well, allowing to specify e.g., that the vehicle should always stay in the right lane, that it should not perform a U-turn if a road sign forbids it, and that speed limits should be obeyed. On top of that, a customer may input so-called safety preferences, again in scLTL: her own rules that she is not willing to disobey, e.g., that a red light should always be respected, or that speeding should not be happening in residential areas. Inspired by our previous works, we propose a function measuring the level of violation of the above three classes of specifications and focus on solving the minimum-violation motion planning problem. The proposed receding horizon solution consists of global long-term routing and local short-term motion planning, which periodically interact and influence each other's outcome. The contributions of the paper can be summarized as follows:

- We formalize the minimum-violation motion planning problem in Pb. 1, allowing to characterize desired behavior of a self-driving car assigned mutually conflicting customer demand and set of road rules.
- We design a receding horizon solution that allows the self-driving car to follow a trajectory that provably minimizes the level of specification violation.
- We demonstrate the applicability of the proposed approach in a simulation case study.

Related work includes planning under infeasible temporal logic specifications [4], [5], [6], where different kind of metrics and strategies are used to measure the level of formula satisfaction and to turn a problem of satisfying an infeasible specification into a problem of optimizing the planning with respect to the chosen metric. Planning for autonomous cars in the context of a mobility-on-demand system was considered e.g., in [7], where a real-time rebalancing policy was developed to maximize the throughput of the system. Similar to [8], [9], in this paper we incorporate limited sensing constraints to develop a reactive sampling-based framework. However, [8], [9] do not consider minimum violation of conflicting temporal constraints. An RRT*-based approach tailored for finding minimum-violation motion plans was proposed in [1], but it did

*This work was partially supported by the NSF under grants NRI-1426907 and CMMI-1400167 at Boston University. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

[1]Cristian-Ioan Vasile, Sertac Karaman, and Daniela Rus are with the Massachusetts Institute of Technology, Cambridge, MA, USA {cvasile, sertac}@mit.edu, rus@csail.mit.edu

[2]Jana Tumova is with KTH Royal Institute of Technology, Stockholm, Sweden tumova@kth.se

[3]Calin Belta is with Boston University, Boston, MA, USA cbelta@bu.edu

not consider sensing, all information was available off-line.

The rest of the paper is structured as follows. In Sec. II we summarize notation and preliminaries. In Sec. III we introduce the hierarchical model of the vehicle in the road network, the three types of specifications, the level of violation and we state our problem. Sec. IV provides a solution and Sec. IV-D discusses its correctness, completeness, and complexity. In Sec. V we provide simulation results. Finally, Sec. VI concludes our work and outlines our ongoing and future work.

II. Preliminaries and Notation

Let $\mathbb{R}$ be the set of real and $\mathbb{N}$ the set of natural numbers. We use $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$, $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$, and $\mathbb{R}_{\geq a} = [a, \infty)$. Given a set S, we denote by 2^{S}, and $|\mathsf{S}|$ the set of all subsets of S, and the cardinality of S, respectively. A sequence of elements from S is called a *word*, and we use w^j to denote the *suffix* $\mathsf{s}_j\mathsf{s}_{j+1}\mathsf{s}_{j+2}\ldots$ starting at the j-th position of a finite or infinite word $w = \mathsf{s}_1\mathsf{s}_2\mathsf{s}_3\ldots$.

A weighted deterministic transition system (WTS) is a tuple $\mathcal{T} = (S, s_{init}, R, W, \Pi, L)$, where S is a finite set of states; $s_{init} \in S$ is the initial state; $R \subseteq S \times S$ is a transition relation; $W : R \to \mathbb{R}_{>0}$ is a weight function; Π is a set of atomic propositions; and $L : S \to 2^{\Pi}$ is a labeling function.

Given that the current state of the system is $s \in S$ at time t, by taking a transition $(s, s') \in R$, the system reaches the state s' at time $t' = t + W\big((s, s')\big)$. A *trace* $\tau = s_1 s_2 s_3 \ldots$ is an infinite sequence of states of $\mathcal{T}$, such that $s_1 = s_{init}$, and $(s_j, s_{j+1}) \in R$, for all $j \geq 1$. The *word* produced by τ is the sequence $w(\tau) = L(s_1)L(s_2)L(s_3)\ldots$.

A syntactically co-safe Linear Temporal Logic (scLTL) formula over alphabet Σ is defined as

$$\varphi ::= \pi \mid \neg\pi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \varphi \,\mathsf{U}\, \varphi,$$

where $\pi \in \Pi$, $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are Boolean operators, and U (until), X (next), and F (eventually) are temporal operators [10].

An scLTL formula is interpreted over infinite words over 2^{Σ}, such as the ones produced by a WTS. The *satisfaction* of an scLTL formula φ by a word $w = w_1 w_2 w_3 \ldots$ over 2^{Σ} is defined through the satisfaction relation $\models$ as follows: $w \models \pi \iff \pi \in w_1$, $w \models \neg\pi \iff \pi \notin w_1$, $w \models \varphi \vee \psi \iff w \models \varphi \vee w \models \psi$, $w \models \varphi \wedge \psi \iff w \models \varphi \wedge w \models \psi$, $w \models \mathsf{X}\varphi \iff w^2 \models \varphi$, $w \models \mathsf{F}\varphi \iff \exists i \geq 1.\, w^i \models \varphi$, $w \models \varphi \,\mathsf{U}\, \psi \iff \exists i \geq 1.\, w^i \models \psi \wedge \forall 1 \leq j < i.\, w^j \models \varphi$.

Although scLTL formulas are defined over infinite words, their satisfaction is decided in finite time [10]. Specifically, a word w over 2^{Π} satisfies φ over Π if it contains a *good prefix* defined as a finite prefix $w_1 w_2 w_3 \ldots w_n$, with the property that $w' = w_1 w_2 w_3 \ldots w_n w'_{n+1} w'_{n+2} \ldots \models \varphi$, for all suffixes $w'_{n+1} w'_{n+2} \ldots$ over 2^{Π}. Given a word w over 2^{Π} and ϕ over Π, a good prefix $w_1 w_2 w_3 \ldots w_n$ of w is *minimal*, if $w_1 w_2 w_3 \ldots w_{n-1}$ is not a good prefix.

The definition of the satisfaction relation is extended to traces of a WTS in the expected way: $\tau \models \varphi$ if and only if $w(\tau) \models \varphi$. A (minimal) *good trace prefix* is the one that produces a (minimal) good prefix.

A deterministic finite automaton (DFA) is a tuple $\mathcal{A} = (Q, q_{init}, \Sigma, \delta, F)$, where Q is a set of states; $q_{init} \in Q$ is the initial state; Σ is a finite alphabet; $\delta \subseteq Q \times \Sigma \to Q$ is a transition function; $F \subseteq Q$ is a set of final states.

A run of a DFA $\mathcal{A}$ over a finite word $w = \sigma_1\sigma_2 \ldots \sigma_n$ is a sequence of states $\rho = q_1 \ldots q_{n+1}$, such that $q_1 = q_{init}$ and $q_{i+1} = \delta(q_i, \sigma_i)$, for all $1 \leq i \leq n$, and it is accepting if $q_{n+1} \in F$. The language of $\mathcal{A}$ is the set of all words that generate accepting runs. A DFA is blocking if δ is a partial function, otherwise it is nonblocking. For any scLTL formula φ over Σ there exists a nonblocking DFA $\mathcal{A} = (Q, q_{init}, 2^{\Sigma}, \delta, F)$, such that the language of $\mathcal{A}$ is the set of all good prefixes of all words that satisfy φ [10].

III. Problem formulation

In this section, we introduce a hierarchical model for route and motion planning for a vehicle tasked with servicing a transportation request expressed as a temporal logic formula, while satisfying road rules and user-defined safety preferences. We introduce the level of violation associated with the delay of servicing of the transportation request and the satisfaction of the road rules and we formalize the problem of minimum-violation motion planning of the vehicle.

A. Hierarchical Model

1) Vehicle model: The vehicle used to service the transportation requests operates in a bounded planar road network $\mathfrak{R} \subset \mathbb{R}^2$ consisting of intersections and roads that connect them. It is modeled as a dynamical system with limited sensing. Formally, the vehicle is defined as a tuple $\mathcal{V} = (f, X, U, h, Sense)$, where $X \subset \mathbb{R}^m$, $U \subset \mathbb{R}^n$, and $\mathfrak{R}$ are the state, control, and work-spaces of the vehicle, $Sense : \mathbb{R}^2 \to 2^{\mathbb{R}^2}$. The dynamics of the model are given by

$$\dot{x} = f(x, u), \quad x(0) = x_0 \tag{1}$$

$$y = h(x) \tag{2}$$

where x_0 is the initial state at time $t = 0$, $f : X \times U \to X$ and $h : X \to \mathbb{R}^2$ are the Lipschitz continuous dynamics and observation (location) functions, respectively. The limited sensing area of the vehicle at location y is given by $Sense(y) \subset \mathbb{R}^2$. The state trajectory under a control policy $u(\cdot)$ is said to be feasible if the location of the vehicle in the planar environment stays inside $\mathfrak{R}$ for all times.

Let Ω be the set of *service region labels* that identify places of interest and are used to specify the transportation request, e.g., a school drop-off point, a shopping mall parking lot, etc. The locations of the regions of interest in the workspace are given by the labeling map $\mathcal{L}_\Omega : X \to 2^{\Omega}$.

2) Road segment model: The road network $\mathfrak{R}$ is decomposed into a set of road segments R. Each road segment $r \in R$ is associated with a compact planar region $\mathfrak{R}_r \subseteq \mathfrak{R}$ in the workspace corresponding to a road lane segment together with its ingoing and outgoing intersections. Hence, two road segments may share an intersection. Except for the ingoing and outgoing one, a road segment does not contain any other intersection. Note that there are two road segments in R for each two-way road, one for each traffic direction.

The space $\mathfrak{R}_r$ is annotated with signs and markings that are used to enforce the rules of the road (e.g., speed limit, stop, and construction signs, lane and intersection delimiters). Let Π denote the set of all signs and markings that are common for

TABLE I: Symbols table.

$\mathcal{V}$	vehicle model
X, U, $\mathfrak{R}$	state, control, and work- spaces
f, h, $Sense$	dynamics, observation (location) and sensing maps
x_0	initial state
Ω, $\mathcal{L}_\Omega$	the set of *service regions* and associated labeling map
$\mathfrak{X}_r$, $\mathfrak{R}_r$	state and work-spaces associated with $r \in R$
Π	set of all signs and markings
$\mathcal{L}_{r,\Pi}$, $\mathcal{L}_{Pi}$	road and network markings and signs labeling maps
$\mathcal{M}_r$	road segment model
S, R	sets of intersections and road segments
L, W	road network labeling and (dynamic) weights maps
ϵ	symbol indicating no service regions on road segment
$x = (p, v)$	state composed of location p and velocity v
$\mathbf{o}_H$, $\boldsymbol{\sigma}_H$	duration output word and output word induced by H
ψ	transportation request
Δ, D	deadline and delay
Θ	set of road rules
θ^a, θ^g	assume and guarantee parts of a road rule
$ReqCompleted$	symbol indicating the end of the request
Φ	safety preference
$(P_\psi, P_\Theta, \beta)$	penalty structure

all road segments. We assume that there exists a minimal set of markings that define right and left lanes, and ingoing and outgoing intersections, denoted by $RightLane$, $LeftLane$, $InIntersection$ and $OutIntersection$, respectively. The regions defined by the markings for road segment $r \in R$ are given by the road labeling map $\mathcal{L}_{r,\Pi} : \mathfrak{X}_r \to 2^\Pi$, where $\mathfrak{X}_r \subseteq X$ is the set of the vehicle's states in the road segment r, i.e., $\mathfrak{X}_r = h^{-1}(\mathfrak{R}_r)$. The road labeling maps must be consistent whenever two road segments $r_1, r_2 \in R$ share an intersection, i.e., $\mathcal{L}^{-1}_{r_1,\Pi}(\{OutIntersection\}) = \mathcal{L}^{-1}_{r_2,\Pi}(\{InIntersection\})$ or $\mathcal{L}^{-1}_{r_2,\Pi}(\{OutIntersection\}) = \mathcal{L}^{-1}_{r_1,\Pi}(\{InIntersection\})$. Note that the regions induced by $\mathcal{L}_{r,\Pi}$ are interpreted as the regions of the road segment r that a sign or marking applies to. For example, $\mathcal{L}_{r,\Pi}$ designates as $construction$ the entire closed-off region of the road segment, and not just the location of the start and end construction signs. $\mathcal{L}_\Pi = \bigcup_{r\in R} \mathcal{L}_{r,\Pi}$ then denotes the assignment of signs and markings over the whole road network $\mathfrak{R}$.

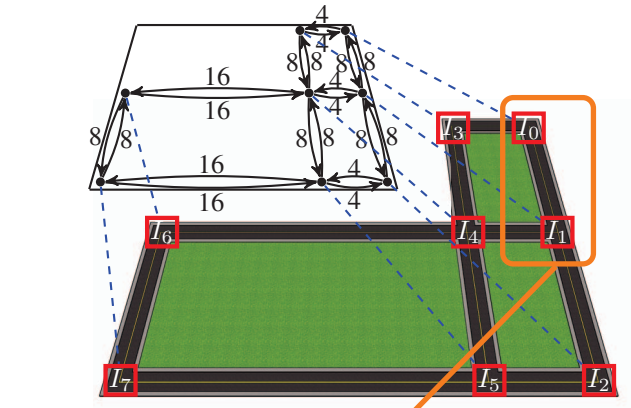
We define the road segment model as a tuple $\mathcal{M}_r = (\mathfrak{R}_r, \Pi, \mathcal{L}_{r,\Pi})$ that captures the traversal of road $r \in R$.

The vehicle has limited sensing and can only perceive the subset of markings in the road segment r that is within its sensing area, i.e., $Sense(y(t)) \cap \mathfrak{R}_r$, where $y(t)$ is the location of the vehicle at time t.

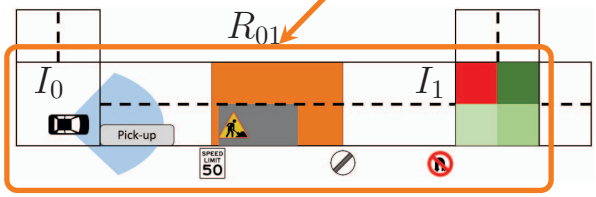
Let r_1, r_2 be two road segments such that the outgoing intersection of r_1 coincides with the ingoing intersection of r_2. The vehicle's state trajectory is said to *switch* from r_1 to r_2 when the vehicle's location enters the outgoing intersection of r_1. Note that the vehicle's location does not change, the switch only affects the sequence of traversed road segments.

3) Road network model: The road network can be viewed as a sparse strongly connected directed graph (S, R) with S representing the sets of intersections and $R \subseteq S \times S$ corresponding to the road segments connecting them, where we slightly abuse the notation for R to denote both the set of road segments and the set of tuples representing the edges of the road network graph. As discussed above, the road segments are labeled with places of interest that may be used to define a transportation request. The road network labeling function $L_\Omega : R \to \Omega \cup \{\epsilon\}$ is obtained from $\mathcal{L}_\Omega$ as follows: $L_\Omega(r) = \mathcal{L}_\Omega(\mathfrak{X}_r)$ for all $r \in R$, where ϵ indicates that the road segment is not associated with any *service regions*. We make the assumption that there is at most one service region per road segment. This assumption is in fact not restrictive as each road segment can be modeled as several ones by introducing additional intersection road segments.

We assume that the vehicle has access to a service that provides information about the traffic conditions in the road network, e.g., INRIX, Google Traffic, Waze. With a slight abuse of notation, estimated travel durations through the road network are captured using a weight function $W : R \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ that assigns to each road segment $r \in R$ at time $t \geq 0$ an estimated (nominal) traversal durations of edge r. Thus, the road network may be modeled as a WTS $\mathcal{T} = (S, s_{init}, R, W, \Omega, L)$ with dynamic weights, where the initial location of the vehicle is in the intersection $s_{init} \in S$. Note that only the traffic information at the current time t_c is available, i.e., $W(\cdot, t_c)$.



(a) Road network.



(b) Road segment model.

Fig. 1: A road network with 8 intersections $I_0, \ldots, I_7$ is shown in (a) in the bottom together with the corresponding WTS in the top. The dots depict the states of the WTS, the arrows represent the transitions labeled with the respective weights, i.e. the estimated travel durations at the current moment. The vehicle is initially located at rest in intersection I_0 and ready to traverse road segment (I_0, I_1), see (b). Intersections are partitioned into 4 quadrants *SharpRight*, *FarRight*, *FarLeft*, and *SharpLeft*, as shown for the intersection I_1 in light, medium, and dark green, and red, respectively. The vehicle's sensing area is a circular arc sector centered at its position and shown in blue. The *pickup* region is marked in light gray, meaning that $L_\Omega((I_0, I_1)) = pickup$ in the WTS in (a) in the top. The road's speed limit, construction zone, and U-turn restrictions are provided through the function $\mathcal{L}_{(I_0,I_1),\Pi}$ illustrated as orange, dark grey, and red regions, respectively. However, due to the limited sensing capabilities, the vehicle perceives only *InIntersection*, *RightLane*, and *LeftLane* markings of (I_0, I_1) at the current moment.

Example 1 *Consider a vehicle in the road network shown in Fig. 1a in the bottom composed of 8 intersections $I_0, \ldots, I_7$. The vehicle dynamics model (Eq.* (1)*) is given via a double integrator vehicle, i.e., $f(x,u) = (0,0,u_1,u_2)$, where the state $x = (p_1, p_2, v_1, v_2)$ is composed of the vehicle's position $p = (p_1, p_2)$ and velocity $v = (v_1, v_2)$, and is controlled by acceleration input $u = (u_1, u_2) \in \mathbb{R}^2$. Thus, $\dot{p}_1 = v_1$, $\dot{p}_2 = v_2$, $\dot{v}_1 = u_1$, and $\dot{v}_2 = u_2$. The observation map that defines the output y (Eq.* (2)*) is trivially $h(x) = p$.*

For the road segment model, we decompose the road network into 20 road segments as prescribed in Sec. III-A.2. The set of all markings is $\Pi = \{$50mphLimit, UnderConstruction, NoUTurn, Under50mph, ConstructionArea, InIntersection, OutIntersection, RightLane, LeftLane, SharpRight, FarRight, FarLeft, SharpLeft$\}$. One of the road segments is illustrated in Fig. 1b together with the road labeling $\mathcal{L}_{(I_0,I_1),\Pi}$. This road labeling is however, not explicitly modeled at this point as it is learned on-the-fly through sensing, i.e. through the function $Sense(y)$. From its current position, the vehicle can observe markings and regions of interest in a circular arc section as shown in Fig. 1b.

The road network modeled is provided through a WTS (see Fig. 1a in the top). The estimated travel times are shown as weights labeling the transitions, and are assumed to be constant over time in this example for simplicity of presentation. The vehicle is initially at the origin (in intersection I_0, which is the initial state of the WTS) and at rest. A customer defines the desired mission using the service regions $\Omega = \{$pickup, bakery, mall, dropoff$\}$ that are mapped to regions in the road network using the labeling map $\mathcal{L}_\Omega$ and further onto the road segments in the WTS using the labeling map L_Ω. Namely, $L_\Omega((I_0, I_1)) =$ pickup, $L_\Omega((I_3, I_0)) =$ bakery, $L_\Omega((I_5, I_7)) =$ mall, and $L_\Omega((I_6, I_7)) =$ dropoff, respectively. Note that the traffic direction is important.

As we will present shortly, the model now allows to state desired specifications, such as that a customer would like to visit a pickup location and then either bakery or mall and that road rules should be met, such as that the vehicle should stay in the right lane or not perform a U-turn when forbidden by a sign.

B. Specification

Let Σ be an alphabet, i.e. a finite set of symbols, x be a trajectory of the vehicle defined by Eq. (1)-(2), and $H : X \to \Sigma$ a labeling function. The *duration output word* $\mathbf{o}_H = (\sigma_1, d_1)(\sigma_2, d_2)\ldots$ corresponding to x with respect to H is defined such that $H(x([t_k, t_{k+1}))) = \sigma_k$ and $\sigma_k \neq \sigma_{k+1}$ for all $k \geq 1$, where $t_{k+1} = t_k + d_k$ and $t_1 = 0$. We denote by $\boldsymbol{\sigma}_H = \sigma_1\sigma_2\ldots$ the output word produced by x. Let ϕ be an scLTL formula over Σ. We say that x *satisfies* ϕ with respect to H, denoted by $x \models_H \phi$, if $\boldsymbol{\sigma}_H \models \phi$. For brevity, we use the notation $x(I) \models_H \phi$ with the meaning that the subword of $\boldsymbol{\sigma}_H$, denoted by $\boldsymbol{\sigma}_H(I)$, corresponding to the time interval I is a good prefix of ϕ.

We say that x *services* ϕ with respect to H if there exists a compliant subword $(\sigma_{i_1}, d_{i_1})(\sigma_{i_2}, d_{i_2})\ldots(\sigma_{i_n}, d_{i_n})$ of the duration output word $\mathbf{o}_H$ and a *compliant good prefix* $\varsigma_{i_1}\varsigma_{i_2}\ldots\varsigma_{i_n}$ of ϕ, where $\varsigma_{i_k} \subseteq \sigma_{i_k}$ for all $k \in \{1, \ldots, n\}$. The corresponding sequence of times $t_{i_1}t_{i_2}\ldots t_{i_n}$ is called the *sequence of service times*. Unlike for the above satisfaction, the labeling function H is not viewed as the assignment of undetachable state properties (such as right lane, left lane etc.) for *servicing* ϕ. Rather, it provides information about available observations that may (but do not have to) be serviced when x passes a state where they hold, e.g., there might be a drop-off area and the vehicle may (but does not have to) stop there.

1) Transportation request: The vehicle is tasked with servicing a request $\psi = (\omega^{start}, \phi, \Delta)$, where ϕ is the route specification expressed as an scLTL formula over the available *service regions* Ω, a start (pick-up) location $\omega^{start} \in \Omega$, and a deadline $\Delta \in \mathbb{R}_{\geq 0}$. The transportation task is thus to service the scLTL formula $\mathsf{F}(\omega^{start} \wedge \phi)$. In other words, the solution to a transportation request is a trajectory x of the vehicle together with a compliant minimal good prefix $\varsigma_{i_1}\varsigma_{i_2}\ldots\varsigma_{i_n}$ of $\mathsf{F}(\omega^{start} \wedge \phi)$ and the corresponding sequence of service times $t_{i_1}t_{i_2}\ldots t_{i_n}$. An optimal solution minimizes the total servicing time $t^{serv} = t_{i_n}$. Ideally, the optimal solution guarantees servicing ϕ before the deadline, i.e., $t^{serv} \leq \Delta$. However, due to traffic conditions, the requirement to satisfy the road rules, the properties of the road network etc., the vehicle might not service the request by the deadline. The delay in servicing the transportation request is $D = t^{serv} - \Delta$.

2) Road rules: Let us introduce a special marking $ReqCompleted \in \Pi$, such that $\mathcal{L}_\Pi^{-1}(\{ReqCompleted\}) = \emptyset$ for all times $t < t^{serv}$ and $\mathcal{L}_\Pi^{-1}(\{ReqCompleted\}) = \mathfrak{R}$ for all times $t \geq t^{serv}$. The rules of the road (RR) are represented as a set Θ of scLTL formulae over Π in reactive form,

$$\theta_j = (\theta_j^a \overset{\text{Re}}{\Rightarrow} \theta_j^g)\,\mathsf{U}\,ReqCompleted \tag{3}$$

for all $\theta_j \in \Theta$, where θ_j^a and θ_j^g are the Boolean assumption and scLTL guarantee formulae, $\overset{\text{Re}}{\Rightarrow}$ (reactive implication) indicates the separation between the two parts of θ_j. We say that a traffic rule θ_j is active if θ_j^a is satisfied. The rules in Θ may become active at any time during the traversal of a road segment.

3) Safety preferences: In addition to the transportation request, the user also provides safety preferences in the form of a set of scLTL formulae Φ over Π. Note that safety preferences can be used to enforce global temporal constraints regarding the violation of traffic rules throughout the whole journey.

Example 2 (Cont.) *Consider the setup in Ex. 1, and the following road rules: 1)* stay in the right lane*; 2)* if the road is under construction, then avoid construction areas*; 3)* if u-turn is not allowed, then don't go to the sharp left quadrant of the intersection*; and 4)* if speed is limited to 50mph, then drive at under 50mph*. The road rules written as an scLTL formula given in Eq.*(3)*, where*

j	θ_j^a	θ_j^g
1	$\top$	$RightLane$
2	$UnderConstruction$	$\neg ConstructionArea$
3	$NoUTurn$	$\neg SharpLeft\mathsf{U}\neg OutIntersection$
4	$50mphLimit$	$Under50mph$

Lastly, consider a transportation request to arrive at a party by $\Delta = 40$ at dropoff *region. The customer is at region $\omega^{start} =$* pickup*, and needs to stop by the* mall *or* bakery *first. The transportation request is $(pickup, \phi_{party}, 40)$, where $\phi_{party} = \mathsf{F}((\text{mall} \vee \text{bakery}) \wedge \mathsf{X}\,\text{dropoff})$ is the scLTL formula associated with the service request. While the customer is in a hurry, she does not want the vehicle to travel above the speed limit and also break other rules of the road. Moreover, at most one illegal U-turn should be performed throughout the journey.*

The scLTL formulae for the safety preferences are $\varphi_1 = \left((\theta_4^a \wedge \neg\theta_4^g) \Rightarrow \bigwedge_{j=1}^{3} \theta_j^a \Rightarrow \theta_j^g\right) \mathsf{U} ReqCompleted$, *and* $\varphi_2 = (\theta_3^v \Rightarrow (\theta_3^v \mathsf{U}(\neg SharpLeft \wedge \theta_3))) \mathsf{U} ReqCompleted$, *where* $\theta_3^v = \theta_3^a \wedge SharpLeft$.

C. Problem Statement

We focus on situations when the transportation request cannot be completed by the deadline without violating the road rules, i.e. we focus on minimum-violation motion planning problems. For the former two types of specifications, we define a level of violation as a function that assigns a penalty to each trajectory of the vehicle and the transportation request and the set of road rules, respectively. However, the user-defined safety preferences are required to be fully satisfied.

1) Transportation request: For the transportation request, the level of specification violation is measured through the delay D as given by the penalty function $P_\psi : \mathbb{R} \to \bar{\mathbb{R}}$.

2) Road rules: For the road rules, the level of violation is defined through the level of unsafety, similarly as in [1]. Each road rule $\theta_j \in \Theta$ is associated with a priority $p_j \in \mathbb{N}$. Intuitively, the level of unsafety is measured as the cumulative time spent satisfying the individual road rule assumptions, but not the guarantees, weighted by the corresponding priority. Formally, given a duration output word $\mathbf{o}_H = (\sigma_1, d_1)(\sigma_2, d_2)\ldots$ corresponding to a trajectory y, the level of unsafety is

$$P_\Theta(x) = \sum_{\theta_j \in \Theta} \left(p_j \cdot \sum_{i \in \{i | \sigma_i \models \theta_j^a \wedge \neg\theta_j^g\}} d_i \right).$$

3) Penalty structure: The relative importance of the delay penalty of the request with respect to the penalties for violation of road rules is denoted by $\beta \in \mathbb{R}_{>0}$. The triplet $(P_\psi, P_\Theta, \beta)$ is called a penalty structure for the traffic network.

Example 3 (Cont.) *In Ex. 1 we consider that rule* θ_2 *is three times more important than* θ_1, *and* θ_3 *and* θ_4 *twice as much. Therefore, the priorities for the rules are 1, 3, 2, and 2, respectively. For simplicity, the penalty function for the delay is the identity. The relative importance* β *is taken as 2, meaning that the level of unsafety is twice as important as the delay penalty.*

Problem 1 *Given a vehicle* $\mathcal{V}$ *operating in the road network* $(\{\mathcal{M}_r\}_{r\in R}, \mathcal{T})$, *a transportation request* ψ, *a set of road rules* Θ, *a set of safety preferences* Φ, *and a penalty structure* $(P_\psi, P_\Theta, \beta)$, *find a control policy* $u(t) : \mathbb{R}_{\geq 0} \to U$ *such that the state trajectory* $x(t)$ *of* $\mathcal{V}$ *is feasible, minimizes the total penalty*

$$P(x) = P_\psi(D) + \beta \cdot P_\Theta(x),$$

and the safety preferences encoded by Φ *are all satisfied.*

IV. Solution

In this section, we propose a receding horizon approach to solve Pb. 1 that leverages recent developments in minimum-violation temporal logic planning. The proposed framework exploits the natural decomposition of Pb. 1 into a global long-term routing problem Pb. 2 and a short-horizon local motion planning problem Pb. 3. Intuitively, a long-term routing plan can be generated that disregards signs and markings since these are *a priori* unknown, and focuses on minimizing the delay penalty based on available routing decisions and the current travel duration estimates for the network's roads. In between intersections, a local planner that takes into account discovered markings and signs, is used to minimize both violation of road rules and transportation delay as captured by the total penalty function, and satisfies the safety preferences. The local planner invokes the routing algorithm to recompute the global plan whenever (1) the computed decision is unavailable or requires a high degree of violation of road rules due to locally sensed signs and markings, and (2) it is not optimal anymore due to changing travel durations estimates.

A minimum-violation routing algorithm is used to generate routes at the road network level modeled as a WTS that minimize the delay penalty associated with the servicing of the transportation request. Routing decisions are made at the intersections of the road network, and the routing algorithm is used to find the next road segment to traverse on the optimal route with respect to the current estimated travel durations. The motion within each road segment (between two intersections) is computed by a local planner. We propose a sampling-based algorithm that incrementally constructs motion plans that minimize the violation of traffic rules and transportation delay, while obeying the customer's safety preferences. The transportation request, rules of the road, and the safety preferences are all converted into DFAs annotated with transition weights that capture the level of violation. Overall, the solution computes controls at each time step to drive the vehicle towards the completion of the transportation request with minimum cost, provided that local motion problem are feasible.

A. Receding Horizon Planner

The receding horizon controller proposed in Alg. 1 is based on the decomposition of Pb. 1 into the following two interconnected, iteratively addressed problems.

Problem 2 (Routing) *Given a WTS* $\mathcal{T}$ *with fixed weights, and transportation request* $\psi = (\omega^{start}, \phi, \Delta)$, *find a trace* τ *that* services $\mathsf{F}(\omega^{start} \wedge \phi)$ *with minimum delay penalty* $P_{Obj}(\Delta)$ *if one exits, otherwise report failure.*

Problem 3 (Local Motion Planning) *Given a vehicle* $\mathcal{V} = (f, \mathfrak{X}_r, U, h, Sense)$ *in a road segment* r, *labeling maps* $\mathcal{L}_\Omega$, *and* $\mathcal{L}_{r,\Pi}$, *a desired service region* ω, *a set of road rules* Θ, *safety preferences encoded as* $\varphi = \bigwedge_{\varphi_\ell \in \Phi} \varphi_\ell$, *and the vehicle's trajectory prefix up to the current moment, find a control policy* $u(\cdot)$ *that induces a continuation of the state trajectory that services the region marked by* ω, *if one exits in* r, *minimally violates the rules of the road, satisfies* φ, *and finishes in the outgoing intersection of the road segment.*

The receding horizon planner Alg. 1 has two stages: an off-line part (lines 1-10) and an on-line part (lines 11-16).

In the off-line stage, the RRT* tree for local motion planning is initialized (line 2) and the initial state is labeled with transportation and road observations (line 3) in the current road segment (line 1). Next, the specifications for the transportation request, safety specification, and each road rule are converted to DFAs, and the initial state x_0 is annotated corresponding to automata states (lines 4-8). The conversion of the transportation

request and safety preference to DFAs is provided using an off-the-self tool such as Spot [11], which we denote by $dfa(\cdot)$ in Alg. 1. The rules of the road are also translated to DFAs, but annotated with transition weights as detailed in Sec. IV-B. Lastly, the cost of x_0 is set to zero, the current state is set to x_0 and added to the solution state trajectory $solution$ (lines 9-10).

In the on-line stage, at each step the current road segment is retrieved (line 12). Based on the current location of the vehicle, Alg. 2 either (1) returns the current road segment r_c in case the vehicle is not at an outgoing intersection (lines 1-6), or (2) it switches to the next road segment that minimized transportation delay (line 9). The routing problem Pb. 2 is solved using the approach proposed in [3], and is denoted in Alg. 2 by $routing(\cdot)$. It takes as input an intersection, the current state in the transportation request DFA, and the current estimated travel times (line 8) and outputs the route in the WTS road network model as well as the next service region to be visited. After the next road segment to be traversed is selected, the receding horizon algorithm Alg. 1 proceeds to relabel and prune the RRT* tree based on the currently available observations inside the vehicle's sensing area (line 13). The tree is relabeled recursively starting from the root in Alg. 3, and states that do not satisfy the safety preference are deleted together with associated subtrees (line 4). The update procedure is described in Sec. IV-C, and uses the labeling functions restricted to the sensing area around the current location $h(x_c)$ (line 2). Note that the root[1] of the RRT* tree is always the current state x_c. The updated RRT* tree then checked for a solution trajectory that extends until the outgoing intersection of the current road segment r (line 14), i.e., $existsSolution(\mathcal{T}_*) = \top$ if and only if $OutIntersection \in \mathcal{L}_{r,\Pi}(leaves(\mathcal{T}_*))$. In case a solution trajectory does not exist, the local planner is called (line 15). The algorithm reports failure if the local planner is unable to generate a motion plan. In case, a solution trajectory exists, the current state x_c is set to the next state on the optimal trajectory, and added to the solution (line 16). The $nextRoot(\cdot)$ procedure also deletes the old root and all subtrees corresponding to the siblings of the new root. The optimal leave state is selected based on the accumulated cost J described in Sec. IV-C.

B. Weighted DFA

Similar to [1], the scLTL formulae for road rules are translated to weighted DFAs that capture the level of violation. The operation is denoted $mva(\theta)$ in Alg. 1 (line 5), and involved two steps: (1) θ is translated to a DFA $\bar{\mathcal{A}}_\theta = (\bar{Q}_\theta, q^\theta_{init}, 2^{\Pi_\theta}, \bar{\delta}_\theta, \bar{F}_\theta)$ using $dfa(\cdot)$, and (2) self-loops and a weight function are added to the $\bar{\mathcal{A}}_\theta$. Let $\mathcal{A}_\theta = (\bar{Q}_\theta, q^\theta_{init}, 2^{\Pi_\theta}, \delta_\theta, \mathcal{W}_\theta, \bar{F}_\theta) \leftarrow mva(\theta)$, where the edge set of $\mathcal{A}$ is $\delta_\theta = \bar{\delta}_\theta \cup \{(q, \sigma, q) \mid \sigma \in 2^{\Pi_\theta} \text{ s.t. } q \neq \bar{\delta}_\theta(q, \sigma)\}$. The weights of $\mathcal{A}$ are defined as $\mathcal{W}_\theta(\tau) = \begin{cases} p & \tau \in \bar{\delta}_\theta \\ 0 & \tau \in \delta_\theta \setminus \bar{\delta}_\theta \end{cases}$, where p is the priority of road rule θ, and $\tau \in \bar{Q}_\theta \times 2^{\Pi_\theta} \times \bar{Q}_\theta$.

As opposed to [1], in this work we use deterministic automata and we do not combine the automata corresponding to road rules into a product automaton. Moreover, the DFAs are defined over subsets of Π denoted by Π_θ that include only the markings involved in the rule. These changes allow the planner to consider only active road rules during computation, and possibly add and remove *a priori* unknown rules on-the-fly in scenarios such as accident sites. The latter is part of ongoing work. Another advantage is that it generates DFAs with less number of states and transitions.

[1] We use standard tree operations to retrieve the parent and children of a state, the root and leaf states, and to delete a sub-tree, and we denote them by pa, ch, $root$, $leaves$, and $deleteSubTree$, respectively.

Algorithm 1: Receding Horizon Planner

Input: $\mathcal{V}$ – vehicle, $(\{\mathcal{M}_r\}_{r\in R}, \mathcal{T})$ – road network, ψ – transportation request, Θ – road rules, φ – safety preferences, $(P_\psi, P_\Theta, \beta)$ – penalty structure
Output: $solution$ – a solution state trajectory or "Failure"

1 $r \leftarrow getRoadSegment(x_0, \bowtie)$ `// off-line`
2 $\mathcal{T}_* \leftarrow (S_* = \{x_0\}, x_0, R_* = \emptyset, W_* = \emptyset, L_* = \emptyset)$
3 $L_*(x_0) = \mathcal{L}_\Omega(x_0) \cup \mathcal{L}_{r,\Pi}(x_0)$
4 **for** $\theta \in \Theta$ **do**
5 $\quad \mathcal{A}_\theta = (Q_\theta, q^\theta_{init}, 2^{\Pi_\theta}, \delta_\theta, \mathcal{W}_\theta, F_\theta) \leftarrow mva(\theta)$
6 $\quad state_\theta(x_0) \leftarrow \delta_\theta(q^\theta_{init}, L_*(x_0))$
7 $\mathcal{A}_\varphi \leftarrow dfa(\varphi)$; $state_\varphi(x_0) \leftarrow \delta_\varphi(q^\varphi_{init}, L_*(x_0))$
8 $\mathcal{A}_\psi \leftarrow dfa(\mathsf{F}(\omega^{start} \wedge \phi))$; $state_\psi(x_0) \leftarrow \delta_\psi(q^\psi_{init}, L_*(x_0))$
9 $J(x_0) \leftarrow 0$
10 $x_c \leftarrow x_0$; $solution \leftarrow (x_c)$
11 **while** $state_\psi(x_c) \notin F_\psi$ **do** `// on-line`
12 $\quad r \leftarrow getRoadSegment(x_c, r)$
13 $\quad prune(root(\mathcal{T}_*), \mathcal{T}_*)$
14 $\quad$ **if** $\neg\, existsSolution(\mathcal{T}_*)$ **then**
15 $\quad\quad$ **if** $\neg\, localPlanner(\mathcal{T}_*, r)$ **then return** *Failure*
16 $\quad x_c \leftarrow nextRoot(\mathcal{T}_*)$; $solution.append(x_c)$
17 **return** $solution$

Algorithm 2: $getRoadSegment(x_c, r_c)$

1 $R_c \leftarrow \{r \in R \mid x_c \in \mathfrak{X}_r\}$
2 **if** $R_c = \{r_c\}$ **then** `// not at intersection`
3 $\quad$ **return** r_c
4 **else** `// intersection: in multiple road segments`
5 $\quad R_c = \{r_p, r^\ell_n\}$ s.t. $r_p = (s', s_c) \wedge r^\ell_n = (s_c, s_\ell), \forall \ell$
6 $\quad$ **if** $\exists \ell \;\; r_c = r^\ell_n$ **then return** r^ℓ_n
7 $\quad$ **else** `// switch road segments: best routing`
8 $\quad\quad W_c \leftarrow W(\cdot, t_c))$
9 $\quad\quad$ **return** $\arg\min_\ell \{C_\ell + routing(s_\ell, state_\varphi(x_c), W_c)\}$, where $C_\ell = \beta \sum W_\theta(state_\varphi(x_c), state_\varphi(s_\ell))$

Algorithm 3: $prune(x, \mathcal{T}_*)$

1 **foreach** $x_{ch} \in ch(x, \mathcal{T}_*)$ **do**
2 $\quad L_*(x_{ch}) = \{\mathcal{L}_\Omega(x_{ch}) \cup \mathcal{L}_{r,\Pi}(x_{ch}) \mid h(x_{ch}) \in Sense(h(x_c))\}$
3 $\quad$ **if** $update(x_{ch})$ **then** $prune(x_{ch}, \mathcal{T}_*)$
4 $\quad$ **else** $deleteSubTree(x_{ch}, \mathcal{T}_*)$

C. Local Planner

The local planner Alg. 4 is responsible for generating motion plans that traverse the road segments selected using the routing algorithm. The method is based on random sampling of the workspace, and uses the following primitive functions: (1) $sample(\mathfrak{R}_r)$ generates a uniformly distributed location in $\mathfrak{R}_r$, (2) $velocity(p)$ returns a velocity at location p (see below for details), (3) $near(x, \mathcal{T}_*)$ returns all states in $\mathcal{T}_*$ that are close to x, i.e., $S_{near} = \{x' \in S_* \mid \|p - p'\|_2 \leq$

Algorithm 4: $localPlanner(\mathcal{T}_\star, r)$

Input: $\mathcal{T}_*$ – current RRT* tree, r – current road segment, and all inputs of Alg. 1
Output: Boolean value – indicates if a solution was found

1 $x_{root} \leftarrow root(\mathcal{T}_*)$
2 **for** $i = 1, \ldots, N_{max}$ **do**
3 $p \leftarrow sample(\mathfrak{R}_r)$
4 $x \leftarrow (p, velocity(p))$
5 $L_*(x) = \{\mathcal{L}_\Omega(x) \cup \mathcal{L}_{r,\Pi}(x) \mid h(x) \in Sense(h(x_{root}))\}$
6 $J(x) \leftarrow \infty$
7 **for** $x' \in near(x, \mathcal{T}_*)$ **do**
8 **if** $steer(x', x)$ **then**
9 $update(x, x')$
10 **if** $x \in S_*$ **then**
11 **for** $x' \in near(x, \mathcal{T}_*)$ **do** // rewire
12 **if** *steer(x, x')* **then**
13 $update(x', x)$
14 **return** $existsSolution(\mathcal{T}_*)$

Algorithm 5: $update(x, x_{pa})$

1 **if** $\delta_\psi(x_{pa}, L_*(x)) = \bowtie$ **then return** $\bot$
2 **else**
3 $C \leftarrow c(x_{pa}, x)$
4 **foreach** $\theta \in \Theta$ **do**
5 $q^\theta_{pa} \leftarrow state_\theta(x_{pa})$; $q^\theta \leftarrow \delta_\theta(q^\theta_{pa}, L_*(x))$
6 $C \leftarrow C + \beta \cdot \mathcal{W}_\theta(q^\theta_{pa}, q^\theta)$
7 **if** $J(x) > C + J(x_{pa})$ **then**
8 **if** $x \notin S_*$ **then**
9 $S_* \leftarrow S_* \cup \{x\}$; $R_* \leftarrow R* \cup \{(x_{pa}, x)\}$
10 **else**
11 $R_* \leftarrow (R* \setminus \{(pa(s, \mathcal{T}_*), x)\}) \cup \{(x_{pa}, x)\}$
12 $state_\psi(x) \leftarrow \delta_\psi(state_\psi(x_{pa}), L_*(x))$
13 $state_\varphi(x) \leftarrow \delta_\varphi(state_\varphi(x_{pa}), L_*(x))$
14 **foreach** $\theta \in \Theta$ **do**
15 $state_\theta(x) \leftarrow \delta_\theta(state_\theta(x_{pa}), L_*(x))$
16 **return** $\top$

$\gamma\sqrt{\log(n)/n}\}$, and (4) $steer(x, x')$ computes a pair $(u(\cdot), T)$ such that $\int_0^T f(x(\tau), u(\tau))\mathrm{d}t = x'$ with $x(0) = x$, and T is minimized. Similar to RRT*, Alg. 4 first attempts to expand the tree $\mathcal{T}_*$ (lines 3-9) and connect the new sample x to the best parent (line 9). Then, the new state is used to improve the cost of the nearby states in $\mathcal{T}_*$ (lines 10-13). Note that the state is labeled with markings and regions of interest within the sensing area associated with the root's location (line 5).

The velocity at the random location p (line 4) is computed based on set of macro-action Act such as change to 0 (stop), 5, . . ., 70 mph. The primitive $velocity(p)$ randomly chooses a macro-action, and returns the corresponding velocity.

The update of DFA states and cost associated with a state x is done by $update(x, x_{pa})$ Alg. 5, where x_{pa} is the potential new parent of x. If the sample satisfies the safety preference ψ (line 1), then the connection cost is computes as a sum of the steering cost $c(x_{pa}, x)$ (line 3) and the road rule violation cost (lines 4-6). If the current cost of x is is larger than the cost induced by considering x_{pa} as the parent of x (line 7), then x_{pa} becomes the parent of x (lines 8-11), and the DFA states associated with x are updated (lines 12-15).

D. Analysis

Theorem 1 *The probability that Alg. 1 returns a control policy u such that the vehicle eventually services the transportation request while satisfying the safety preferences approaches one as number of samples per road (N_{max}) segment approaches infinity, if a solution exists. Moreover, at each moment the computed routing and motion plan is optimal with respect to the current state and available information, i.e. it gives a solution to Pb. 1 at each time t.*

Proof: (Sketch) The proof of the first part follows from the asymptotic optimality of RRT*. If a solution exists, then there exists a motion plan within each road segment, and the sampling-based algorithm Alg. 4 finds it with probability 1 as N_{max} tends to infinity. The second part follows from the optimality of the local planner and the routing algorithm, see [3] for the optimality proof of the routing algorithm. ■

E. Complexity

Theorem 2 *The complexity of the off-line phase and each step of the on-line phase of Alg. 1 are $O(|\Theta|2^{max_{\theta\in\Theta}|\theta|} + 2^{|\psi|} + 2^{|\phi|} + |S|\log|S|)$ and $O(|S|\log|S| + |\Theta| \cdot |\mathcal{T}_*| + |\Theta| \cdot |\mathcal{T}_*|\log|\mathcal{T}_*|)$, respectively.*

Proof: The 3 terms of the off-line phase correspond to translation of the scLTL formulae to DFAs, and the last term is the complexity of the routing algorithm. The first term of the on-line phase is again the complexity of the routing procedure used to determine the next road segment to traverse. The second term corresponds to the pruning procedure. The final term corresponds to the local planner, where $|\mathcal{T}_*|$ represents the number of states of $\mathcal{T}_*$. Note that the complexity of local planner does not increase with respect to $|\mathcal{T}_*|$, because we use deterministic automata to represent the specifications. ■

Moreover, the decomposition of the road network into segments aids in the computation of motion plans, since it induces smaller local planning problems within each segment that overall have lower complexity than computing a global motion plan over the entire road network directly.

V. Computational Experiments

The algorithms presented in this paper were implemented in Python2.7 using the LOMAP package [12]. All examples were ran on a Intel Core i7 computer with 2.4GHz processor and 8GB RAM under Ubuntu 14.04.

In the following, we present the a case study, where a vehicle operating in the road network shown in Fig. 1a and 2e must service a transportation request to "pickup the customer at *pickup*, visit either the *mall* or *bakery*, and drop-off her off at *dropoff*" within 40 time units. The vehicle must obey the rules of the road in a minimum-violating manner, and the safety preferences unconditionally. An extended description of the setup is presented in Ex. 1, 2, and 3. We report computed routing plans, the vehicle's overall trajectory and associated cost, and execution time of the on-line phase of the receding horizon solution Alg. 1 to evaluate its performance with respect to Pb. 1.

We ran Alg. 1 for a vehicle with sensing radius 1, and with $N_{max} = 2000$ samples per road segment. Fig. 2a shows the tree

TS computed from the initial state, where the construction area is not within the sensing radius. The receding horizon algorithm implements the motion prescribed by the local motion plan, and Fig. 2b shows the vehicle after a few steps before sensing the *construction area*. When the *construction area* enters the vehicle's sensing field-of-view, the planner recomputes a tree TS 2b that avoids the region as shown in 2d, where the vehicle has overtaken the *construction* (obstacle) region and is moving towards the *pickup* regions. Note that the vehicle violates the requirement θ_1 to drive on the right in order to drive pass the *construction area*, and the behavior is consistent with the priorities of the road rules, i.e., $p_1 = 1$ for θ_1 compared to $p_2 = 3$ for θ_3.

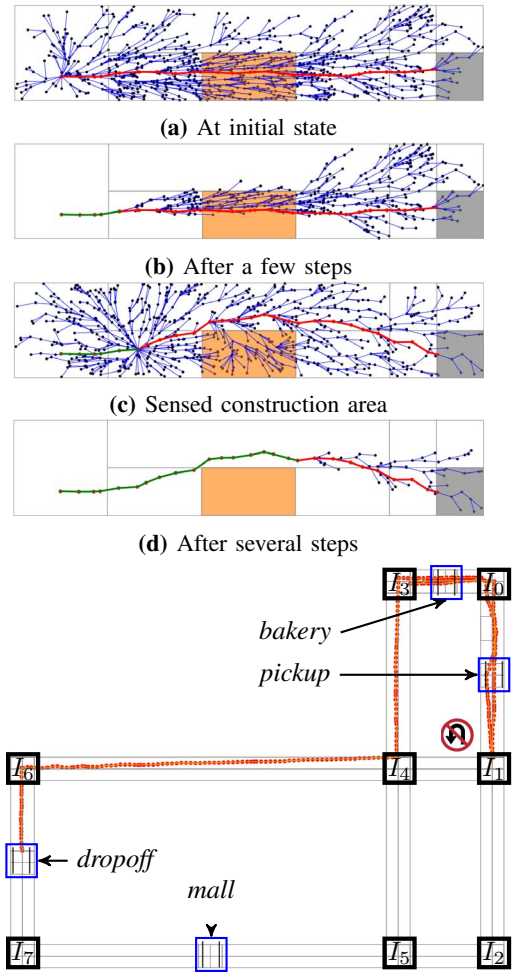
The overall trajectory of the vehicle that services the transportation request from Ex. 2 is shown in Fig. 2e projected on the road network. At the initial state, an optimal route for the vehicle is $(I_0, I_1, I_4, I_5, I_7, I_6, I_7\langle dropoff\rangle)$ with an estimated duration of 48, assuming that *dropff* is at the middle of road segment (I_6, I_7). However, after arriving at intersection I_1, the vehicle learns that the estimated travel times for (I_1, I_4) and (I_5, I_7) changed to 6 and 26, respectively. Thus, the new optimal route becomes $(I_0, I_1, I_0, I_3, I_0, I_3, I_4, I_6, I_7\langle dropoff\rangle)$ with an estimated duration of 56. In order to implement the new route, the planner violates the rule θ_3 by performing a forbidden U-turn at intersection I_1. However, the safety preferences are met as the vehicle never performs a forbidden U-turn again nor exceeds 50mph on speed constrained stretches of the road segments. The actual delay incurred by following the motion plan was 28.027 time units with a total cost of 56.114.

The on-line phase of Alg. 1 was performed 319 times with an average execution time of 2.545 seconds (std. 6.044) and 13.52 minutes for the entire request. Note that computed local plans are reused within road segments, and recomputing is performed when the vehicle senses new markings and signs.

VI. Conclusions

We have studied the problem of motion planning for an autonomous vehicle that is given a conflicting set of customer demands and road rules specified in a temporal logic. A systematic receding approach to derive the minimum-violation motion plan has been designed that builds on integrated high-level routing in the road network and low level RRT* motion planning along road segments. An illustrative computational experiment demonstrates the use of the proposed approach.

Future work involves several research directions from supporting a finer characterization of the state space to accommodating multiple demands and vehicle sharing to considering other vehicles and dynamic elements in the traffic network.



(a) At initial state

(b) After a few steps

(c) Sensed construction area

(d) After several steps

(e) Overall trajectory projected in the road network

Fig. 2: The figures show the generated tree TS in road segment (I_0, I_1), a–d, and the overall state trajectory of the vehicle servicing the transportation request projected onto the planar road network, e. The orange and grey regions represent the construction area and the pickup location, respectively. The state and transitions of the RRT tree are marked in black and blue, the computed path to the target regions *pickup* is shown in red, and the trajectory up until the current location is green. Note that the *service regions*, e.g., *mall*, *bakery*, *pickup* and *dropoff*, can only be serviced if road segments are traversed in the correct direction.

References

[1] L. Reyes Castro, P. Chaudhari, *et al.*, "Incremental sampling-based algorithm for minimum-violation motion planning," in *IEEE Conference on Decision and Control*, 2013, pp. 3217–3224.

[2] J. Tumova, G. C. Hall, *et al.*, "Least-violating control strategy synthesis with safety rules," in *International Conference on Hybrid Systems: Computation and Control*, Philadelphia, PA, USA, 2013, pp. 1–10.

[3] J. Tumova, S. Karaman, *et al.*, "Least-violating planning in road networks from temporal logic specifications," in *ACM/IEEE International Conference on Cyber-Physical Systems*, 2016, pp. 1–9.

[4] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the minimal revision problem of specification automata," *The International Journal of Robotics Research*, 2015.

[5] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, Feb. 2015.

[6] M. Maly, M. Lahijanian, *et al.*, "Iterative Temporal Motion Planning for Hybrid Systems in Partially Unknown Environments," in *Int. Conference on Hybrid Systems: Computation and Control*, 2013.

[7] E. F. M. Pavone, S. L. Smith and D. Rus, "Robotic load balancing for mobility-on-demand systems," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[8] C. Vasile and C. Belta, "Sampling-Based Temporal Logic Path Planning," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[9] ——, "Reactive Sampling-Based Temporal Logic Path Planning," in *IEEE Int. Conference on Robotics and Automation*, Hong Kong, 2014.

[10] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[11] A. Duret-Lutz, A. Lewkowicz, *et al.*, "Spot 2.0 – a framework for LTL and ω-automata manipulation," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2016.

[12] A. Ulusoy, S. L. Smith, *et al.*, "Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints," *Int. Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.