When to Terminate: Path Non-existence Verification Improves Sampling-based Motion Planning

Jesper Karlsson[1], Anastasiia Varava[1], Cristian-Ioan Vasile[2],
Sertac Karaman[3], Danica Kragic[1], Daniela Rus[3] and Jana Tumova[1]

***Abstract*— In this work, we combine a sampling based motion planner with path non-existence verification. We show that, using our approach, it is possible to: 1) provide termination criteria to sampling-based motion planners, based on the problem specific constraints; 2) generate a method for rejection sampling that can adapt to the scenario, for instance, leveraging the relative priority between constraints. Furthermore, we describe a language-guided sampling technique based on IRRT*, that utilizes the results from the path non-existence verification procedure to reduce runtime of the underlying motion planner. The approach is studied using route and motion planning for an autonomous vehicle that aims to service transportation tasks while constrained by rules of the road. We evaluate the proposed approach using a set of real-life inspired traffic scenarios. Finally, we show that the resulting runtime of a motion planner with path non-existence verification is significantly shortened when compared to the same motion planner with traditional termination criteria, such as a resource or time limit.**

I. Introduction

Motion planning for safety critical application requires predictable and stable runtime performance. Although sampling-based planners are very popular [1], they lack formal termination criteria and infeasibility guarantees. Termination criteria can improve (a) the safety and explainability of planning by showing when and why sampling should stop, and (b) the efficiency by avoiding unnecessary computation. Moreover, methods for termination decisions must take into account rich problem specific contexts. In this work, we show how to use path non-existence verification to improve sampling-based motion planning under temporal logic rules.

For path finding in continuous spaces, asymptotically complete motion planners are a popular tool, as the computational complexity of complete motion planners make practical implementations infeasible. Sampling-based motion planners such as probabilistic roadmaps (PRM) and rapidly exploring random tree star (RRT*) [2] are probabilistically complete. Extensions to RRT*, achieving real-time performance have been proposed by for instance, Arslan et al. and Otte et al. in RRT$^\#$ [3] and RRT^X [4], respectively. There has been work on exploring the theoretical properties of sampling-based motion planners [5]. However, termination criteria are generally given in terms of a time limit or a set number of samples drawn. This can be detrimental to the performance of the motion planner. For instance, when no solution exists in the current configuration, the planner will continue to search for a solution until the time limit is reached. Vice versa, if a path is not found within the given time limit, no conclusions can be drawn regarding the existence or non-existence of a path. Attempts has been made to alleviate this problem, and in [6] the authors attempt to provide a confidence value for the finite-time solution, i.e. a probability on the existence of a solution in a certain environment.

In contrast to our previous work proposing asymptotically optimal route and motion planning for autonomous driving with road rules [7], we strive here to direct the search for plans in order to outperform the original algorithm in terms of convergence rate and solution cost. This can be seen in relation to works such as [8], where the authors use reachability analysis for the computation of reachable sets that provide the planner with information regarding the drivable area. However, the computation of reachable sets for more complex dynamics (i.e. non-linear) is difficult. Similarly, works such as [9][10] are inherently unsuitable for providing termination criteria for sampling-based approaches as they require an evaluation of trajectories, rather than regions of the workspace. Furthermore, these approaches do not support Linear Temporal Logic (LTL) constraints. Therefore, we propose the use of the path non-existence verification approach described in [11], as it does not require the user to provide the dynamics model.

Several works attempt to integrate path non-existence verification in sampling-based motion planners. For instance, the authors in [12] present a path non-existence verification algorithm based on adaptive cell decomposition of the configuration space. A similar approach was implemented in [13], where the author construct parts of the collision space using alpha shapes. The main limitation of these approaches is their computational complexity, as even a two-dimensional object in a planar setting presents a computational challenge. These methods have not been generalized to higher dimensions. A more efficient path non-existence algorithm that can be generalized to higher dimensions was proposed in [11]. This algorithm constructs an approximation of the configuration space of an object in order to analyze its connectivity, and provide provably correct path non-existence verification. Tajvar et al. [14], employed this method in

[1]Jesper Karlsson, Anastasiia Varava, Danica Kragic and Jana Tumova are with the division of Robotics, Perception and Learning at KTH Royal Institute of Technology, Stockholm, Sweden. They are also affiliated with Digital Futures. {jeskarl, varava, dani, tumova}@kth.se

[2]Cristian-Ioan Vasile is with Lehigh University, Bethlehem, PA, USA crv519@lehigh.edu

[3]Sertac Karaman, and Daniela Rus are with the Massachusetts Institute of Technology, Cambridge, MA, USA sertac@mit.edu, rus@csail.mit.edu

combination with abstraction refinement in order to provide motion planning for systems with non-holonomic dynamics and non-trivial geometry.

In this work, we propose to combine a sampling-based motion planner [7] with a method for provable path non-existence verification [11]. Our approach utilizes a sampling procedure based on the Informed RRT* [15] (IRRT*), which focuses the drawn samples to regions of interest specified by the path non-existence verification method.

To evaluate the approach, we consider the problem where a vehicle operates in a road network aiming to complete A-to-B transportation tasks, while obeying a set of road rules. The road rules are encoded in the syntactically co-safe fragment of linear temporal logic (scLTL), which is rich enough to express a variety of complex constraints. The road-rules that we model are: 1) to avoid the left lane and bus lanes; 2) to maintain the speed limit; 3) to avoid construction zones and parked vehicles. As it might not be possible to strictly enforce all road rules throughout the vehicles' trip, the route and motion planner in this work is *least-violating*. As such, potential solutions can violate road rules in order to complete the transportation task. However, the planner aims to find the solution that minimizes said violation with regards to the duration spent violating a road rule as well as the priority (importance) of that road rule.

The contributions can be summarized as follows: 1) We introduce a termination criterion for a least-violating sampling-based motion planner based on path non-existence verification; 2) We design a road rule relaxation and re-routing procedure triggered by a proof of path non-existence for a particular road segment; 3) We guide sampling in the motion planner according to the degree of road rules satisfaction; 4) We propose a variant of the IRRT* algorithm, that takes into account the active road rules in a scene. We show that these enhancements, can significantly reduce the computational runtime and improve the overall solution quality.

II. Preliminaries and Notation

Let $\mathbb{R}$ and $\mathbb{N}$ denote the set of real and natural numbers, respectively. Given a set $\mathcal{S}$, let $2^{\mathcal{S}}$ denote the set of all subsets of $\mathcal{S}$. A *word* $w = w_1 w_2 w_3 \dots w_k$ is a sequence of elements from $\mathcal{S}$.

A syntactically co-safe Linear Temporal Logic (scLTL) formula is defined over an alphabet Π as follows:
$\varphi \ ::= \ \pi \mid \neg\pi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \varphi \,\mathsf{U}\, \varphi$, where $\pi \in \Pi$ are atomic propositions, $\neg$, $\wedge$ and $\vee$ are Boolean operators, and U (Until), X (neXt), F (eventually, in the Future) are temporal operators [16]. scLTL formulae are interpreted over infinite words over 2^{Π}, but can be captured using deterministic finite-state automata (DFA), because their satisfaction can be decided in finite time. There are readily available tools for translation of scLTL to DFA, such as: ltl2ba [17], ltl3ba [18] and spot [19].

Least-Violating RRT* (LV-RRT*) is an algorithm developed in [7] that aims to return a motion plan minimizing the *level of violation* – the (weighted) cumulative time spent satisfying the assumptions, but not the guarantees – of a set of (weighted) scLTL formulas in a reactive form $(\theta_j^a \overset{\text{Re}}{\Rightarrow} \theta_j^g)\ \mathsf{U}\ goal$, where θ_j^a is a boolean assumption formula, θ_j^g is an scLTL guarantee formula, $\overset{\text{Re}}{\Rightarrow}$ is a (reactive) implication, and $goal \in \Pi$ is an atomic proposition indicated completion of the specification.

LV-RRT* starts by translating each scLTL formula into a DFA and enhances it with self-loops and a weight function, so that the resulting weighted DFA captures the level of violation of all words w.r.t. the scLTL formula. Next, LV-RRT* proceeds similarly as RRT*, however, starting from the initial state, it randomly samples the workspace, connects new samples to the best parent in the tree, and rewires the tree. In addition, LV-RRT* keeps track of the current state of weighted DFAs and the rewiring is based on the weights of the DFAs, in order to ensure asymptotic optimality of the level of violation of the found motion plan.

III. Problem Statement

In this work, we follow the problem setup we developed in [7]. In particular, we consider an autonomous vehicle operating in a traffic network, servicing transportation requests, while constrained by rules of the road. In this work, we only consider the static obstacles. The road network is represented as a hierarchical model consisting of a road network graph, the road segment the AV currently traverses, as well as its dynamics (see Fig. 1). The model allows for route planning in the network graph to satisfy the transportation task, while at the same time support lower level motion planning in the road segment to obey by the rules of the road. The hierarchical model is described in detail in [7] and here, we pinpoint the parts relevant to this work.

1) Vehicle model: We define a vehicle ν as a tuple $\nu = (X, U, \mathfrak{R}, f, x_0, h, Sense, \mathcal{L}, \Pi)$, where $X \subset \mathbb{R}^m$, $U \subset \mathbb{R}^n$ and $\mathfrak{R} \subset \mathbb{R}^2$ are the state, control and workspaces. $\mathfrak{R}$ corresponds to the road network, which consists of road lanes and intersections. The vehicle's dynamics are as follows:

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

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

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 continuous dynamics and the observation function. Let $Sense : \mathbb{R}^2 \to 2^{\mathfrak{R}}$ be the vehicle's sensing region, which defines the region where road signs and markings can be seen by the vehicle. Let Π denote the set of road signs and markings that annotate the road network. Furthermore, let $\mathcal{L} : X \to 2^{\Pi}$ denote the road labeling map that assigns labels from Π to the vehicle states in the sensing area $Sense(y)$. These labels are used to specify where certain road rules hold. This means that $\mathcal{L}(t)$ defines the bounds of each road rule in a road segment, e.g. where a specific speed limit holds. In this work we assume that we have access to perfect sensors, so that the labelled regions are fully deterministic. We define the duration output word $\boldsymbol{o} = (\sigma_1, d_1)(\sigma_2, d_2) \dots (\sigma_k, d_k)$, corresponding to a trajectory x, such that $\mathcal{L}(x[t_i, t_{i+1}]) = \sigma_i$ and $\sigma_i \neq \sigma_{i+1} \forall i \geq 0$, where $t_{i+1} = t_i + d_i$ and $t_1 = 0$.

2) Road segment model: The road network consists of a set of road segments $\mathfrak{R}_s \subseteq \mathfrak{R}$. Each road segment $\mathfrak{R}_s$ contains a road lane segment r as well as an outgoing intersection r_o together with an ingoing intersection r_i. Let $span(r)$ denote the bounds of region where the road lane segment, or intersection, is defined. We further define $span(r)_w$ as the width of the road lane segment, and $span(r)_l$ as its length.

3) Task: The vehicle task is defined as follows:

$$\Theta \cup g_{drop-off} \equiv (\theta_j^a \overset{\text{Re}}{\Rightarrow} \theta_j^g) \cup g_{drop-off}, \tag{3}$$

where $\theta_j^a \in \Theta^a$, $\theta_j^g \in \Theta^g$ and $\overset{\text{Re}}{\Rightarrow}$ are boolean assumption formulas, and a reactive implication, respectively. Θ is the set of road rules that are considered and $g_{drop-off} \in \mathfrak{R}$ is the goal location of the vehicle. A traffic rule θ_j is considered active if the assumption formula θ_j^a is satisfied. Intuitively, the reactive implication can be understood as adding restrictions on the AV based on what road signs or markings are seen in the sensing region. We say that a road rule is *critical* if it needs to be strictly enforced throughout the run in order to ensure safety. Such a road rule is specified as $\theta_j^c \subseteq \Theta$. Furthermore, each road rule, $\theta_j \in \Theta$, has a corresponding *priority*, $p_j \in \mathbb{N}_\infty$, which signifies the relative importance of satisfying the road rule. The priority of critical rules is ∞. Each road rule is translated into DFA, annotated with transition weights corresponding to the priority, in order to capture the violation cost.

4) Trajectory cost: The cost of the trajectory considers the duration of a task is weighed against the accumulated cost of violating road rules throughout the run. In other words, given a duration output word, $\boldsymbol{o} = (\sigma_1, d_1)(\sigma_2, d_2)\dots(\sigma_k, d_k)$, associated to a trajectory x of vehicle ν, we define the level of violation as:

$$P(x) = \sum_{\theta_j \in \Theta} \left(p_j \sum_{k \in \{k | \sigma_k \models \theta_j^a \wedge \neg \theta_j^g\}} d_k \right). \tag{4}$$

Finally, the total cost of a trajectory x is specified as

$$J(x) = \sum_{k=1}^{|\boldsymbol{o}|} d_k + \beta P(x), \tag{5}$$

where β is a constant relaying the relative importance of safety versus the duration of a task.

The motion planning problem that we address throughout this work is closely related to the problem defined in previous work [7], in which the goal is to find the optimal trajectory $x^\star$ with respect to $J(x)$ (Eq. 4). The goal of this work is to present an approach that can provide termination criteria for sampling-based motion planners by utilizing path non-existence verification. Furthermore, we construct a method for rejection sampling based on the information provided by the path non-existence verification.

IV. Problem Solution

In this section we give an overview of path non-existence verification problem. We then present how path non-existence informs route and motion planning and finally, we discuss completeness and complexity of the overall algorithm.

A. Algorithm overview

The approach is illustrated in Fig. 1 and outlined in Alg. 1. First, an initial route through the road network is computed (line 1) [7], [20]. As long as the goal destination is not reached, the algorithm follows the planned route, and repeatedly plans the motion in the next road segment (line 2). In each iteration, we check whether a path exists given the current road segment $\mathfrak{R}_s$ and information from $Sense(y)$ by running the path non-existence verification algorithm from [11] (line 3). Path non-existence is performed by checking the existence of a path for every set of road rules in 2^Θ. The algorithm returns $\theta^\star$, which is interpreted as the maximal road rule combination that can be strictly enforced throughout the road segment, and $\tilde{P}$ which is the estimated cost of violating $\theta^\star$ (see Sec. IV-C). The road segment is *blocked* if there does not exist a path in the free space obeying by all critical rules θ_j^c. The road segment is considered *sub-optimal* if the estimated cost $\tilde{P}$ of traversing the road segment would trigger higher overall cost then when following an alternative route through the road network. Both blocked and sub-optimal road segment triggers rerouting (see Sec. IV-C.1). In the motion planning algorithm (least-violating road-rule-informed RRT*, or LV-RR-IRRT*), we bias sampling so as to strictly enforce the road rule combination $\theta^\star$ (line 12) (see Sec. IV-C.2).

Algorithm 1: Least-violating route and motion planner with path non-existence queries

```
   input: V - vehicle; R - Road network; Θ - road rules;
1  Routing(V, R, Θ);
2  while goal destination not reached do
3      (θ*, P̃) ← pathExist(R_s);
4      if Blocked or Sub − optimal then
5          R ← R \ R_s;
6          Re-Routing(s, R);
7          if ¬route then
8              return Failure
9          end
10         LV-RR-IRRT*(V, R, Θ);
11     else
12         LV-RR-IRRT*(V, R_s, θ*);
13 end
14 return Success
```

B. Path non-existence

The problem of verifying path non-existence can be considered as dual to path planning. It deals with the question whether or not an object (e.g. robot or vehicle) can move between a start and a goal configuration. In this case, we consider the path non-existence problem, formulated in [11], of computing the connected components of the free space for 2D and 3D workspaces. Although the algorithm is general and can handle objects with no rotational symmetries, in this paper it is sufficient to use its simplified

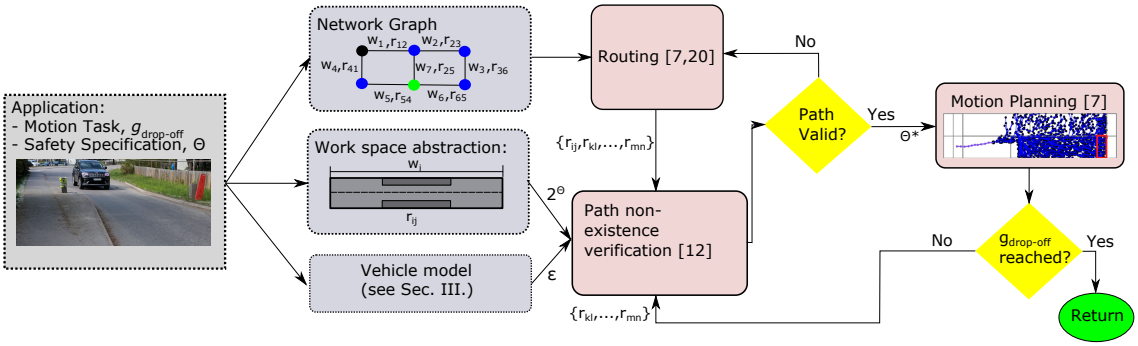


Fig. 1: Illustration of the algorithm proposed in Alg. 1. The blue boxes (dotted border), represents the the hierarchical model (see Sec. III). The red boxes (solid border), represents the steps combining the two approaches, least-violating motion planning from [7] and path non-existence verification from [11].

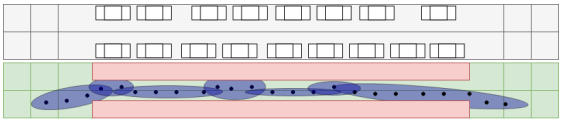


Fig. 2: The picture in the top represents a scenario on a road with parked vehicles. The red region denotes regions pruned away with path non-existence queries. The green region shows region from which we could sample, and finally the blue region represents the informed set from which we actually sample.

version as the model of the vehicle is invariant with respect to rotations. Therefore, its configuration space has only two dimensions, which results in path existence checks that take only 1 - 3 milliseconds.

Let $\mathcal{E} \subset \mathbb{R}^2$, $\mathcal{C}$ denote the finite footprint of the vehicle and its configuration space, respectively. Let $\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_n \subset \Re$ denote set of regions in the workspace corresponding to obstacles and regions forbidden by the critical rules of the road. The connectivity checker takes the segment of the road under consideration as input, and verifies whether the vehicle can move from the start to the end goal. Every time the local planner is called, we construct an approximation of $\mathcal{C}^{free}_{i,j,...,m}$ with respect to the constraints $\mathcal{R}_0 = \{\mathcal{R}_i, \mathcal{R}_j, ..., \mathcal{R}_m\}$ that are specified. Once that is done, we check whether the start and the goal regions are located in the same connected component of the free space. If they are not, this means that there is no path between them that would obey all constraints under consideration.

C. Path non-existence-informed planning

In general it may not possible to satisfy all road rules Θ throughout the run; with this in mind, we look for a subset of Θ that can be satisfied simultaneously and represent the most critical road rules. To identify them, we consider the following cost estimation, based on Eq. (4):

$$\tilde{P}(\Theta') = \sum_{\theta_j \in \Theta} p_j \; g(\mathcal{L}_{\theta_j}), \tag{6}$$

where $g(\mathcal{L})$ is any admissible heuristic function that estimates the duration of violation of the road rules d_k, and $\Theta' \subseteq \Theta$ is a subset of road rules that can be strictly enforced throughout the run. This subset is provided by the path non-existence verification, as it allows us to determine whether a solution exists that never violates the road rules $\theta \in \Theta'$. The purpose of Eq. 6, is to provide a metric for comparing different combinations of hard vs. soft constraints on the rules of the road. In this work we let $g(\mathcal{L}) = min(A_l, A_w)/v_{max}$, where A_l, A_w are the length and width of the region $\mathcal{L}_{\theta_k}$ where the kth road rule holds and v_{max} is the maximum velocity allowed in the road segment. From this, the road rule configuration that is strictly enforced throughout the road segment is determined by:

$$\theta^\star = \underset{\theta_j \in \Theta}{\operatorname{argmax}} \, \tilde{P}(\Theta'). \tag{7}$$

In other words, we strictly enforce the set of road rules that provide the highest estimated penalty. The chosen road rule combination is added to the list of *critical* road rules, and is strictly enforced. Due to the guarantees provided by the path non-existence verification, sample rejection can be utilized on these regions without a loss to the quality of the resulting trajectory, thus maintaining the completeness of the motion planner [7] while improving the overall efficiency.

1) Rerouting: The primary effect that path non-existence guarantees has on the planner is, whether or not it is necessary to find an alternative route through the road network. A road segment is *blocked* if $\theta^c = \emptyset$, i.e. when we can not ensure the safety-critical road rules according to the path non-existence algorithm. A road segment is *sub-optimal* when the union of the regions induced by $\mathcal{L}_{\Theta \setminus \theta^\star}$ satisfies the following relations:

$$span(\mathcal{L}_{\Theta \setminus \theta^\star})_w \geq span(r)_w - \epsilon, \tag{8}$$

$$\tilde{P}(\theta^\star) \geq D' - \sum_{k=1}^{|o|} d_k, \tag{9}$$

where D' is the duration to the transportation task resulting from following an alternative path through the road network, and *span* denotes a width of a region. In other words, we say that a road segment is blocked if the region induced by the road rules $\Theta \setminus \theta^\star$ is larger than the width of the road with some margin ϵ, and the alternative cost is larger than the duration resulting from following the alternative path. Both blocked and sub-optimal segments trigger rerouting.

2) Sample Biasing: Informed RRT* (IRRT*), first introduced in [15], is an extension to RRT* constraining the search space in the sampling procedure so as to improve the convergence rate. As our problem is concerned with not only minimizing path length, but also the violation of the rules of the road Θ, using the standard version of IRRT* is too conservative. We propose *Road rule-informed RRT** (LV-RR-IRRT*), a variant of IRRT* that takes into consideration active road rules.

IRRT* maintains an *informed subset* X_f in the form of a hyperspheroid around a proposed solution. The informed region is defined as:

$$X_f = \{x \in X \mid ||x_0 - x||_2 + ||x - x_k||_2 \leq c_{best}\}, \quad (10)$$

where x_k is the final state and c_{best} is the cost of the current best solution under consideration. The road rule-informed version we propose in this work functions in a similar manner, with one key difference. The informed subset X_f does not consist of a single hyperspheroid, rather a set of hyperspheroids, each defined over the segment of the solution where the markings of a separate road rule holds, i.e. $\mathcal{L}(x_{i,i+T}) = \Pi_\theta$. Once an informed region is defined, we sample such that: $Pr(x \in X_f) = 1 - \epsilon$ and $Pr(x \in X \setminus X_{\theta^\star}) = \epsilon$. This ensures that the solution does not converge towards a local minimum.

Example 1 *In Fig. 2, a vehicle traverses a road with parked cars on both sides of the road. The forbidden regions in this example are labelled $LeftLane$ and $ParkedCars$, respectively. Initially, path non-existence checks are sent for the road rule combinations. This results in $\theta^* = \{true \overset{\text{Re}}{\Rightarrow} \neg\, ParkedCars\}$, which corresponds to the red region in Fig. 2. Rejection sampling is performed so that no samples are drawn from this region. The motion planner attempts to find an initial solution using uniform sampling within the region $X \setminus X_{\theta^*}$. Once an initial solution is found, we construct the informed region X_f as the set of hyperspheriods (blue in Fig. 2) defined over the segment of the trajectory x where a certain road rule, or label corresponding to that road rule $\mathcal{L}(x_{i,i+T})$, holds.*

D. Completeness and Complexity

Following the combined properties employed in this work, if our algorithm reports path non-existence then it is guaranteed to be correct. Otherwise, the algorithm will operate under the asymptotically completeness properties of the least-violating planner.

If our vehicle model was not invariant with respect to rotations, it may happen that our path non-existence verification algorithm would report that there is a path while in reality the start and the goal regions are disconnected [11]. The reason for this is that the algorithm constructs an over-approximation of the free space; hence two disconnected regions of the free space can be over-approximated to the extent that they appear to be connected in the approximation. This does not happen in the present work, as the vehicle is assumed to have a disk-shaped safety footprint, and therefore we do not take different orientations into account when constructing the free space approximation.

Let $|S|$ be the number of intersections in $\mathfrak{R}$, $\mathcal{T}^\star$ the RRT* tree, n and m be the number of balls in the object's and obstacle's spherical representation, and b the maximum number of balls in each pair of connected components. k is the number of pairwise intersections of the balls.

Theorem 1 ***(Complexity)*** *The complexity of the algorithm is: $\mathcal{O}(|\Theta|2^{\max_{\theta\in\Theta}|\theta|} + 2^{|\Theta|} + |S|\log|S|)$ and $\mathcal{O}(|S|\log|S| + |\Theta||\mathcal{T}^\star|\log|\mathcal{T}^\star| + n^2m^2 + b\log^3(b) + k)$, for the offline and online stages respectively.*

Proof: The offline stage consists of translating the scLTL formulae to DFA as well as the routing procedure. The first term of the online stage corresponds to the routing procedure, the second and third to the pruning and local planning [7]. The final two terms of the online stage corresponds to the path non-existence verification procedure [11]. ■

The main benefit comes in the scenario where no solution exists, in these cases the complexity of the online stage is reduced to: $\mathcal{O}(|S|\log|\ S|\ + n^2m^2 + b\log^3(b) + k)$, which corresponds to the routing and path non-existence verification procedures. As we can see, the complexity in this case is completely independent on $|\mathcal{T}^\star|$, and scales purely with respect to the complexity of the environment.

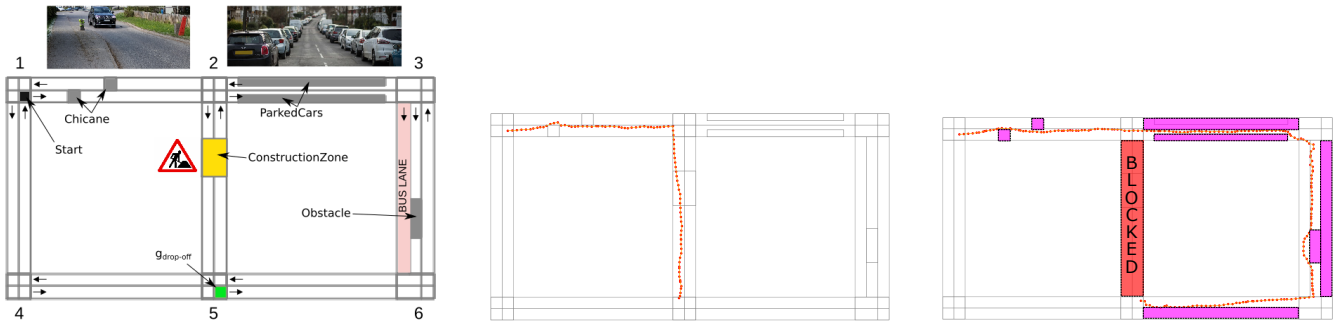
V. Simulations

Here we present the results from running the motion planner on a road network inspired by real-life road conditions. The road network studied in this work is depicted in Fig. 3a. The pick-up location is illustrated in black, and the drop-off location in green. The first segment of interest (intersection 1-2) is a chicane. The second road segment (intersection 2-3) is one where traversing the road is difficult due to parked cars on both sides of the road. The third one (intersection 2-5), is a road that is closed, thus forcing cars to take a different route. The final segment of interest (intersection 3-6), is one with a bus lane and a blocked middle lane, which requires the planner to chose whether to enter the bus lane or the left lane. For all cases, we want the vehicle to maintain a speed limit of 50 mph and stay in the right lane whenever possible. For more details on the encoding of the road rules we refer to our previous work [7].

The experiments have been performed using both samples and time limit as resource constraint, in this case 4000 samples and 60 seconds. We compared the original least-violating motion planner (LV-RRT*) [7] with the path non-existence extension, with and without the road rule-informed sampling procedure described in Sec. IV-C.2 (LV-RR-IRRT*). The data presented in this section is the mean result over 20 runs of each method.

A. Rerouting Results

In Fig. 3b, we observe how the LV-RRT* is limited by the initial route, while in Fig. 3c we see how the path non-existence allows the planner to detect a blockage and



(a) (b) (c)

Fig. 3: (a) A road network. The vehicle is to move from the initial configuration in the black marked region, to the green region. (b) Results without path non-existence verification. The resulting trajectory violates a critical road-rule. (c) Results with path non-existence verification. Purple (dashed border) indicates those regions that the path non-existence verification guarantees can be strictly enforced, and are therefore rejected during sampling.

re-route. The least-violating planner from [7] provides a costly solution by entering the construction zone, since the chosen route is based on initial knowledge of the network. Conversely, the extended version incorporates the new information, determines that this segment is *blocked* and reroutes. This results in a path that is safer, in terms of road rule violation, but longer.

B. Sample Bias Results

An alternative approach to implement rerouting criteria, is to impose a highest allowed cost on the resulting solution. We implemented this strategy in order to illustrate the improved performance when using path non-existence verification.

Fig. 4a outlines the comparison of path cost against time in segment 1-2. We can see that there is no clear computational benefit of running the extension on a single segment, that also has a simple workspace. This is consistent with the complexity analysis provided in Sec. IV-D. In fact, we can see that the LV-RR-IRRT* actually performs worse on aggregate than both LV-RRT* and path non-existence without LV-RR-IRRT*. The reason for this is that the LV-RR-IRRT* approach represents an inherently greedy search where exploitation of a known solution is elected over exploration of the workspace. This can be detrimental in simpler workspaces such as the one in segment 1-2. However, the performance is significantly improved in the complex parts of the route, and, overall, the runtime is improved.

Fig. 4b outlines the evolution of the length of the solution trajectory over time, from the time an initial solution is found until the time limit. This is an important metric as the length of the resulting trajectory has a significant impact on the trajectory duration, and therefore its overall cost. This case illustrates the final segment (Segment 6-5) in the mission and we can see how the LV-RR-IRRT* reduces the overall runtime. By utilizing the LV-RR-IRRT* approach, we can represent the same information, but with a smaller tree. This becomes especially important over longer runs, as it reduces the time necessary to perform graph processing, such as updating information as a new object has been sensed in the scene. Fig. 4c and Fig. 5 illustrates the tree size over time for the first segment and the full run, respectively. We can see that the LV-RR-IRRT* approach consistently provides smaller trees. However, the solutions are comparable, and in some cases better than uniform sampling and sample rejection using path non-existence verification, which is illustrated in Fig. 4b. Thus the algorithm is able to overcome difficult traffic situations much faster.

VI. Conclusion

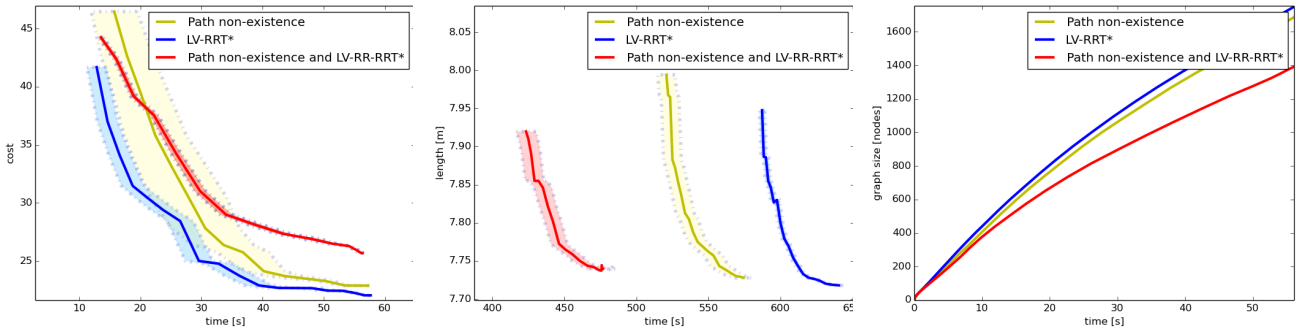
We have shown that by combining a sampling-based motion planner with a path non-existence verification algorithm, we can produce a more efficient motion planner, in particular when no solution exists as well as in more complex workspaces. We have also provided a biasing approach using the information provided by the verification procedure. The LV-RR-IRRT* allows us to express the same information in a smaller tree. This in turn provides an efficiency improvement in cases where graph processing is necessary. For instance, when new information is introduced to the scene. The performance improvement has been shown theoretically and illustrated using a real-life inspired traffic scenario.

Currently we consider a simple vehicle model with a safety footprint that is invariant with respect to rotations, such that its configuration space is two-dimensional. In the future we will generalize our approach to higher dimensions. This will allow us to consider less conservative vehicle safety footprints, as well as joint configuration spaces of multiple vehicles. In our experiments we have not considered dynamic environments (for instance, involving other vehicles in the scene). In future work we plan on exploring how scenario classification on the mapping layer can provide information on when the scene has changed enough to necessitate replanning, and thereby allow our approach to be extended to dynamic environments.

VII. Acknowledgments

We would like to thank Christian Pek for his valuable and constructive suggestions.

This work is supported in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, the Swedish Research Council (VR) (project no. 2017-05102), by the NSF



(a) (b) (c)

Fig. 4: (a) Path cost vs. execution time (i.e. runtime of the entire pipeline, including planning and execution of the trajectory) in segment 1-2 in road network defined in Fig. 3. (b) Path length vs. execution time in segment 6-5. The LV-RRT* (blue) algorithm is delayed since it exhausted the resource constraints in segment 2-5. The path non-existence with LV-RR-IRRT* is faster than both LV-RRT* and path non-existence without LV-RR-IRRT*. (c) Graph size vs execution time in first segment. Due to the rejection sampling, the resulting tree is smaller, which increases the efficiency of any graph processing methods.

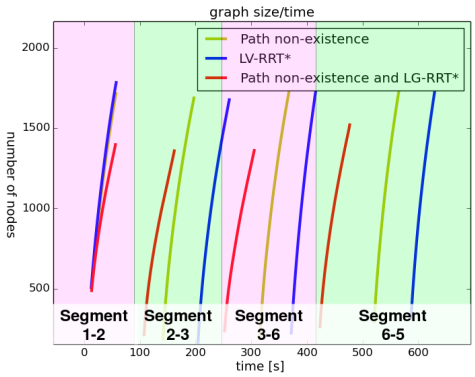


Fig. 5: Graph size vs. execution time over the full run. In each new segment, the tree is reset. The change in colour illustrates when a new segment is reached by any method. Segment label is listed in the bottom of the figure.

Grant 1723943, and the Office of Naval Research (ONR) Grant N00014-18-1-2830.

References

[1] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[2] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[3] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *IEEE International Conference on Robotics and Automation*, pp. 2421–2428, IEEE, 2013.

[4] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[5] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[6] A. Dobson and K. E. Bekris, "A study on the finite-time near-optimality properties of sampling-based motion planners," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1236–1241, 2013.

[7] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scltl motion planning for mobility-on-demand," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1481–1488, IEEE, 2017.

[8] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[9] S. Söntges and M. Althoff, "Determining the nonexistence of evasive trajectories for collision avoidance systems," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 956–961, IEEE, 2015.

[10] S. Sontges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1891–1897, IEEE, 2018.

[11] A. Varava, J. F. Carvalho, F. T. Pokorny, and D. Kragic, "Caging and path non-existence: a deterministic sampling-based verification algorithm," *Robotics Research*, pp. 589–604, 2020.

[12] L. Zhang, Y. J. Kim, and D. Manocha, "A simple path non-existence algorithm using C-obstacle query," in *Algorithmic Foundation of Robotics VII*, pp. 269–284, Springer, 2008.

[13] Z. Mccarthy, T. Bretl, and S. Hutchinson, "Proving Path Non-existence Using Sampling and Alpha Shapes," *IEEE International Conference on Robotics and Automation*, pp. 2563–2569, 2012.

[14] P. Tajvar, A. Varava, D. Kragic, and J. Tumova, "Robust motion planning for non-holonomicrobots with planar geometric constraints," in *The International Symposium on Robotics Research, Hanoi, Vietnam*, pp. 1–16, 2019.

[15] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, 2014.

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

[17] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *International Conference on Computer Aided Verification*, pp. 53–65, Springer, 2001.

[18] T. Babiak, M. Křetínskỳ, V. Řehák, and J. Strejček, "Ltl to büchi automata translation: Fast and more deterministic," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 95–109, Springer, 2012.

[19] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 — a framework for LTL and ω-automata manipulation," in *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, vol. 9938 of *Lecture Notes in Computer Science*, pp. 122–129, Springer, Oct. 2016.

[20] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, p. 17, IEEE Press, 2016.