Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control

Kristoffer Bergman, Oskar Ljungqvist and Daniel Axehill

Bergman, K., Ljungqvist, O., Axehill, D., (2020), Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control, *Transactions on Intelligent Vehicles*.
https://doi.org/10.1109/TIV.2020.2991951

Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control

Kristoffer Bergman, Oskar Ljungqvist and Daniel Axehill

***Abstract*—This paper presents a unified optimization-based path planning approach to efficiently compute locally optimal solutions to optimal path planning problems in unstructured environments. The approach is motivated by showing that a lattice-based planner can be cast and analyzed as a bilevel optimization problem. This insight is used to integrate a lattice-based planner and an optimal control-based method in a novel way. The lattice-based planner is applied to the problem in a first step using a discretized search space. In a second step, an optimal control-based method is applied using the lattice-based solution as an initial iterate. In contrast to prior work, the system dynamics and objective function used in the first step are chosen to coincide with those used in the second step. As an important consequence, the lattice planner provides a solution which is highly suitable as a warm-start to the optimal control step. This proposed combination makes, in a structured way, benefit of sampling-based methods ability to solve combinatorial parts of the problem and optimal control-based methods ability to obtain locally optimal solutions. Compared to previous work, the proposed approach is shown in simulations to provide significant improvements in terms of computation time, numerical reliability and objective function value.**

***Index Terms*—Control and Optimization, Motion Planning, Autonomous Vehicles**

I. Introduction

THE problem of computing locally optimal paths for autonomous vehicles such as self-driving cars, unmanned aerial vehicles and autonomous underwater vehicles has recently been extensively studied [1], [2]. However, the task of finding (locally) optimal motions for nonholonomic vehicles in narrow, unstructured environments is still considered difficult [3]. In this paper, optimal path planning is defined as the problem of finding a feasible and collision-free path from the vehicle initial state to a desired goal state, while a specified performance measure is minimized. This path is then intended to be used as a reference for a path-following controller, such as the ones described in [2], [4], [5].

There exist several methods to generate optimized paths for autonomous vehicles. One common approach is to use B-splines or Bezier curves for differentially flat systems, either to smoothen a sequence of waypoints [6], [7] or as steering functions within a sampling-based motion planner [8]. The use of these methods are computationally efficient since the model of the system can be described analytically. However, these methods are not applicable to non-flat systems, such as, e.g., many truck and trailer systems [9]. Furthermore, it is difficult to optimize the maneuvers with respect to a general performance measure.

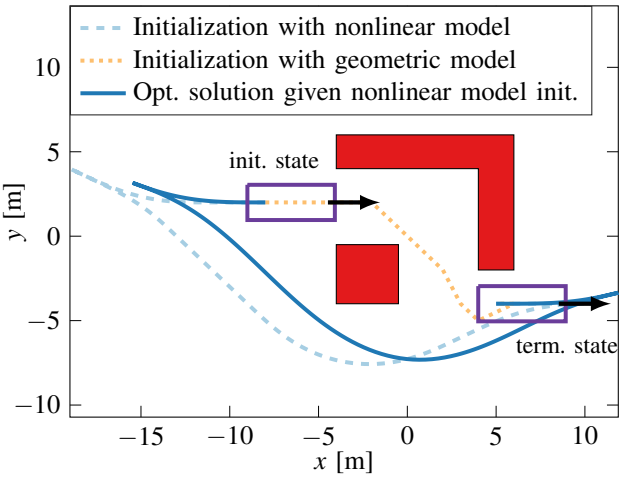

Fig. 1: Feasibility issues. The solution from a path planner based on a simplified geometric model (dotted yellow) provides an initialization from where it is impossible for an OCP solver to find a feasible solution. On the other hand, an initialization based on the full nonlinear model (dashed blue) enables reliable convergence.

Another popular method is to formulate the problem as an optimal control problem (OCP). One approach is to use mixed integer formulations, which can be computationally demanding to solve [10]. Another widely used approach is to use direct methods for optimal control, where the problem is transformed to a nonlinear program (NLP). Due to non-convex constraints introduced by obstacles and the nonlinear system model, a proper initialization strategy is crucial to converge to a good local optimum [11]. A straightforward initialization strategy is to use linear interpolation [12], this can however often lead to convergence issues in cluttered environments [3], [13]. A more sophisticated initialization strategy is to use the solution from a sampling-based path planner. In previous work in the literature, the vehicle models used by the path planner for initialization are simplified; either they disregard the vehicle model completely (geometric planning) [1], [14], or partially (respecting only kinematic constraints) [3], [15]. Using a geometric path planning initialization is shown in [16] to cause problems for vehicles with nonholonomic constraints (illustrated in Fig. 1). Furthermore, initializations based on simplified models will in general be infeasible in the actual OCP to be solved, and potentially not homotopic to a feasible solution [17]. Finally, the objective function in the sampling-based planner can only consider states that are represented in the chosen simplified model description, which might result in a path far from a local minimum.

This work was partially supported by FFI/VINNOVA and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

K. Bergman, O. Ljungqvist and D. Axehill are with the Division of Automatic Control, Linköping University, Sweden {kristoffer.bergman, oskar.ljungqvist, daniel.axehill}@liu.se

One commonly used approach for solving path planning problems is to apply sampling-based path planners, which are either based on random or deterministic exploration of the state space [1]. Many sampling-based algorithms with random exploration are based on the standard Rapidly exploring Random Tree (RRT) algorithm [18], which was originally developed for solving motion planning problems without considering dynamical nor nonholonomic constraints. One example where this algorithm has been modified to account for these types of constraints is RRT* for dynamical systems [19], [17]. In these methods, the orginal RRT algorithm is extended with a rewiring step in each expansion of the search tree. With this modification, the RRT* algorithm is shown to be asymptotically optimal [19], i.e., it will almost surely find the optimal solution as time approaches infinity. A limitation of these methods is that an OCP is required to be solved between each pair of states that are selected for connection online, a process which can be very computationally demanding for a general nonlinear system [17].

A popular deterministic sampling-based path planner is the so-called lattice-based path planner, which uses a finite set of precomputed motion segments online to find a resolution-optimal solution to the path planning problem [20]. A major benefit with this method is that efficient graph-search algorithms can be used online such as A* [21] and ARA* [22], making it efficient to use online [20], [23]. However, since the lattice-based planner uses a discretized search space, the computed solution can be noticeably suboptimal and it is therefore often desirable to improve this solution [8], [15].

The main contribution of this paper is to combine a lattice-based path planner and optimal control in a novel way to compute locally optimal solutions to advanced optimal path planning problems. While the combination of these methods has been studied in previous work, this work significantly improves the overall result when these methods are combined. Motivated by arguments from bilevel optimization, *all* steps within the proposed approach use the *same* system model and objective function, which is in contrast to previous work where, e.g., a simplified model and/or objective function is used in the lattice-based planner. The resulting benefits of the proposed approach are that the lattice-based planner provides a better initialization for the second optimal control step in terms of quality and feasibility. This is confirmed in a simulation study where the proposed approach is used to solve challenging path planning problems for both cars and truck and trailer systems. The proposed approach results in a significant reduction in overall computation time, more reliable convergence of the OCP solver and generally improved solutions compared to state-of-the art techniques based on simplified initialization strategies or lattice-based planners alone.

II. Problem formulation

In this section, the continuous optimal path planning problem is formulated as an OCP. Furthermore, we pose a commonly used approximation of the original problem in terms of a lattice-based path planner.

A. The optimal path planning problem

In this work, path planning problems for nonlinear systems in the form

$$\frac{\mathrm{d}x}{\mathrm{d}s} = x'(s) = f_{q(s)}(x(s),u(s)), \quad x(0) = x_{\text{init}},$$

are considered. Here, $s > 0$ is defined as the distance traveled by the system, $x \in \mathcal{X} \subseteq \mathbf{R}^{n_x}$ is the state vector, $u \in \mathcal{U} \subseteq \mathbf{R}^{n_u}$ is the continuous control input and x_{init} represents the initial state. There is also a discrete input signal $q(s) \in \mathcal{Q} = \{1,2,\ldots,N\}$ which enables the selection between N modes of the system. The system mode determines the vector field $f_q \in \mathcal{F}$ that describes the current equation of motion [24]. The system mode can for example represent the direction of motion (which is the main use in this paper). However, the results presented in Section III-IV also hold for a set $\mathcal{F}$ representing a more general switched dynamical system. One such example is morphing aerial vehicles [25]. Furthermore, the system should not collide with obstacles, where the obstacle region is defined as $\mathcal{X}_{\text{obst}}$. Thus, in path planning problems, the state space is constrained as $x \in \mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obst}}$.

The optimal path planning problem can now be defined as the problem of computing feasible paths in $x(\cdot)$, $u(\cdot)$ and $q(\cdot)$ that move the system from its initial state $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ to a desired terminal state $x_{\text{term}} \in \mathcal{X}_{\text{free}}$ while a performance measure J is minimized. This problem can be posed as the following OCP:

$$\begin{aligned} \underset{x(\cdot),u(\cdot),S_f,q(\cdot)}{\text{minimize}} \quad & J = \int_0^{S_f} \ell(x(s),u(s),q(s))\mathrm{d}s \\ \text{subject to} \quad & x(0) = x_{\text{init}}, \quad x(S_f) = x_{\text{term}}, \\ & x'(s) = f_{q(s)}(x(s),u(s)), \\ & x(s) \in \mathcal{X}_{\text{free}}, \\ & q(s) \in \mathcal{Q}, \quad u(s) \in \mathcal{U}. \end{aligned} \tag{1}$$

Here, the decision variable S_f is the total length of the path and $l(x,u,q)$ forms the cost function that is used to define the performance measure J. Note that the cost function is allowed to depend on the system mode, which enables the possibility of associating each system mode with a unique cost function.

Example 1: Consider a car-like vehicle described by a kinematic bicycle model [1] with augmented state vector $x(s) = (\bar{x}(s),\alpha(s),\omega(s))$, where $\bar{x}(s) = (x_1(s),y_1(s),\theta_1(s))$. Here, (x_1,y_1) denotes the position of the car's rear axle, θ_1 is the car's orientation, α is the front-wheel steering angle and ω is the steering angle rate. The vehicle model is

$$\begin{aligned} \bar{x}'(s) &= q\left(\cos\theta_1(s), \sin\theta_1(s), \frac{\tan\alpha(s)}{L_1}\right)^T, \\ \alpha'(s) &= \omega(s), \quad \omega'(s) = u_\omega(s), \end{aligned}$$

where u_ω is the continuous control signal which represents the steering angle acceleration. Hence, the standard kinematic bicycle model is augmented to account for rate and acceleration constraints on the steering angle α. Furthermore, L_1 is the wheel-base of the car and $q \in \{1,-1\}$ is the discrete decision variable which represents the motion direction, where $q = 1$ implies forward motion and $q = -1$ backward motion.

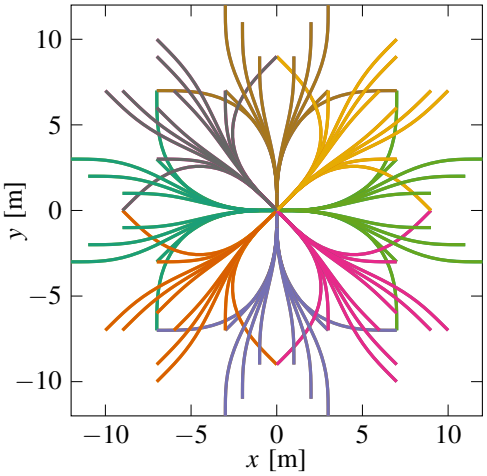

Fig. 2: Examples of motion primitives in forward motion computed offline for the system in Example 1. These can be used to solve path planning problems in the form (3) online. Each color represents an initial heading $\theta_{1,d}$ that is valid in $\mathcal{X}_d$.

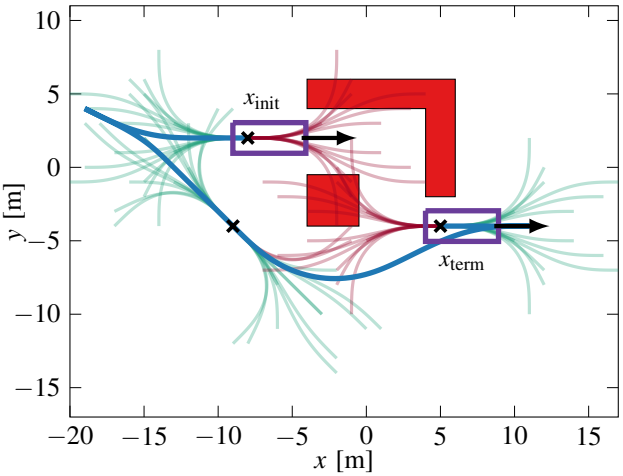

Fig. 3: A solution to a minimum path length problem computed by combining motion primitives online. The applicable motion primitives at three different states along the solution is also shown. The infeasible motions (in collision) are marked in red, while the feasible ones are marked in green.

Due to that the problem of solving (1) contains the combinatorial aspects of selecting the system mode and the route to avoid obstacles, as well as the continuous nonlinear model of the system, finding a feasible and locally optimal solution is a difficult problem. Hence, approximate methods aiming at finding feasible, suboptimal solutions are commonly used [1], where the lattice-based path planner provides one alternative.

B. Lattice-based path planner

The main idea with a lattice-based path planner is to restrict the controls to a discrete subset of the available actions (motion primitives) and as a result transform the optimization problem in (1) into a discrete graph-search problem. In this paper, the so-called state-lattice methodology will be used. The methodology is mainly suitable for position-invariant systems, since then motion primitives need only be computed from states with a position in the origin, and can then be translated to the desired position when applied online [26].

The construction of a state lattice is performed offline and can be divided into three steps. First, a desired state space discretization $\mathcal{X}_d$ is defined that represents the reachable states in the graph. After the discretization has been selected, the connectivity in the graph is chosen by selecting which neighboring states to connect. Finally, the set of motion primitives $\mathcal{P}$ is constructed by computing N_m motion segments (for example using an OCP solver as in [20], [5]) needed to connect the neighboring states, without considering obstacles. A motion primitive $m \in \mathcal{P}$ is defined as

$$m = \big(x(s), u(s), q\big) \in \mathcal{X} \times \mathcal{U} \times \mathcal{Q},\ s \in [0, S], \tag{2}$$

and represents a feasible path of length S in a fixed system mode $q \in \mathcal{Q}$ that moves the system from an initial state $x(0) \in \mathcal{X}_d$ to a final state $x(S) \in \mathcal{X}_d$ by applying the control inputs in $u(\cdot)$. Fig. 2 shows examples of motion primitives in forward motion ($q = 1$) from different initial headings for the car-like vehicle in Example 1. Once the set of motion primitives has been computed, the original path planning problem (1) can be approximated by the following discrete OCP:

$$\begin{aligned}
\underset{\{m_k\}_{k=0}^{M-1},\, M}{\text{minimize}} \quad & J_d = \sum_{k=0}^{M-1} L_u(m_k) && \text{(3a)}\\
\text{subject to} \quad & x_0 = x_{\text{init}}, \quad x_M = x_{\text{term}}, && \text{(3b)}\\
& x_{k+1} = f_m(x_k, m_k), && \text{(3c)}\\
& m_k \in \mathcal{P}(x_k), && \text{(3d)}\\
& c(x_k, m_k) \in \mathcal{X}_{\text{free}}. && \text{(3e)}
\end{aligned}$$

The decision variables are the number of phases M and the applied sequence of motion primitives $\{m_k\}_{k=0}^{M-1}$. Note that the combinatorial aspect of selecting the system mode sequence $\{q_k\}_{k=0}^{M-1}$, and the total length S_f in (1) is implicitly encoded in the selection of motion primitives. The state transition equation in (3c) describes the successor state $x_{k+1} \in \mathcal{X}_d$ after m_k is applied from $x_k \in \mathcal{X}_d$ and (3d) ensures that m_k is selected from the set of applicable motion primitives at x_k. The constraint in (3e), ensures that the vehicle does not collide with any obstacle when m_k is applied from x_k. Finally, the stage cost in the objective function in (3a) is given by

$$L_u(m) = \int_0^S \ell(x(s), u(s), q)\mathrm{d}s. \tag{4}$$

Fig. 3 illustrates how a minimum path length planning problem is solved using the lattice-based approximation in (3) for the car-like vehicle defined in Example 1.

The approximation in (3) enables the use of graph-search algorithms online, making it efficient to solve [20], [5]. The solution is also guaranteed to be feasible. On the downside, the solutions from a lattice-based planner often suffer from discretization artifacts [8], [15], making it desirable to smoothen the state-lattice solution. Other limitations with graph-search methods are that it is only possible to plan from and to states within the selected state-space discretization, and that these methods are resolution complete [1]. These limitations can be alleviated by using multi-resolution lattice approaches in, e.g., narrow passages [27].

III. Bilevel optimization

In this section, the two problems in (1) and (3) will be related by rewriting the original problem formulation (1) into a bilevel optimization problem (BOP) [28]. It will be shown that this new *equivalent* formulation of the problem in (1) allows for an insightful interpretation of the standard lattice solution methodology. In particular it will be used to connect the methodology to parametric optimization, highlight suboptimality properties, and discuss the choice of objective function used at different parts of the lattice-based framework.

A BOP is an optimization problem where a subset of the variables are constrained to be an optimal solution to another optimization problem called the lower-level problem. Analogously, the problem on the first level is called the upper-level problem. A general BOP can be written as [28]

$$\begin{aligned} \underset{y,z}{\text{minimize}} \quad & F(y,z) \\ \text{subject to} \quad & (y,z) \in \Upsilon \\ & z \in \arg\min_{z} \{f(y,z) : (y,z) \in \Omega\}. \end{aligned} \tag{5}$$

Here, $y \in \mathbf{R}^{n_y}$ and $z \in \mathbf{R}^{n_z}$ are the upper and lower-level decision variables, respectively. Furthermore, $F(x,y)$ and $f(x,y)$ represent the upper and lower-level objective functions, respectively, and

$$\begin{aligned} \Upsilon &= \{(y,z) \mid G_i(y,z) \leq 0,\ i \in \{1,\dots,C\}\}, \\ \Omega &= \{(y,z) \mid g_j(y,z) \leq 0,\ j \in \{1,\dots,D\}\}, \end{aligned}$$

represent the upper and lower-level feasible sets, which are represented by C upper (G_i) and D lower-level (g_j) inequality constraints, respectively. Typically, a subset of the optimization variables in the upper-level problem are considered as parameters to the lower-level problem. Seen from the upper-level problem, the optimality requirement of the lower-level problem is in general a non-convex constraint. Comparably simple examples of bilevel problems, e.g., where the problems on both levels are quadratic programming problems, can be solved by representing the solution to the lower-level problem by, e.g., encoding the Karush-Kuhn-Tucker (KKT) conditions using mixed-integer optimization [28] or explicitly representing the lower-level solution parametrically using a solution obtained from parametric programming [29]. It will be shown in this work that the lattice-based planner can be interpreted as a way of solving a bilevel formulation of (1) using the latter alternative, i.e., representing the lower-level solution explicitly as a (sampled) parametric solution.

A. Bilevel optimization problem reformulation

It will now be shown how the path planning problem in (1) can be reformulated as a BOP. Let $L_u(m)$ from (4) represent the upper-level objective function and introduce lower-level cost function $\ell_l(x,u,q)$. Assume that

$$L_u(m) = \int_0^S \ell_l\big(x(s),u(s),q\big)\mathrm{d}s. \tag{6}$$

After dividing the path planning problem in (1) in M path segments where along each one the system mode is kept constant, it is possible to cast it as an *equivalent* bilevel (dynamic) optimization problem in the form

$$\begin{aligned} \underset{\{x_k^0,x_k^f,q_k,m_k\}_{k=0}^{M-1},\, M}{\text{minimize}} \quad & J_u = \sum_{k=0}^{M-1} L_u(m_k) \\ \text{subject to} \quad & x_0^0 = x_{\text{init}}, \quad x_{M-1}^f = x_{\text{term}}, \\ & x_k^0 = x_{k-1}^f, \quad q_k \in \mathcal{Q}, \\ & m_k \in \underset{(x_k,u_k,S_k)}{\arg\min}\ (8), \end{aligned} \tag{7}$$

where the initial state x_k^0, terminal state x_k^g and system mode q_k for phase k are the upper-level optimization variables considered as parameters to the lower-level optimization problem. The constraints given by $x_k^0 = x_{k-1}^g$ ensure that the path is continuous between adjacent path segments. Furthermore, the corresponding lower-level optimization problem in (7) can formally be specified as the following multi-parametric OCP (mp-OCP)

$$\begin{aligned} J_l^*(x_k^0,x_k^f,q_k) = \underset{x_k(\cdot),u_k(\cdot),S_k}{\text{minimize}} \quad & \int_0^{S_k} \ell_l\big(x_k(s),u_k(s),q_k\big)\mathrm{d}s \\ \text{subject to} \quad & x_k(0) = x_k^0,\ x_k(S_k) = x_k^f, \\ & x_k'(s) = f_{q_k}(x_k(s),u_k(s)), \\ & x_k(s) \in \mathcal{X}_{\text{free}}, \\ & u_k(s) \in \mathcal{U}. \end{aligned} \tag{8}$$

where the initial and terminal states x_k^0, x_k^f and the system mode q_k are considered as parameters from the upper-level problem. Note the similarities between this problem and (1). Here, the main difference is that the system mode is fixed by the selection of the upper-level parameter q_k and the path length S_k is typically shorter than S_f (since $S_f = \sum_k S_k$).

Above, it was assumed that the objective functions are related as in (6), which was necessary in order for the equivalence between (1) and (7) to hold. An alternative is to select the objective functions in the two levels more freely in a way that does not satisfy (6), with the price of breaking the equivalence between (1) and (7). If such a choice is still made, the solution to (7) with (8) will in general no longer be an optimal solution to (1). However, the use of different objective functions allows in practice for a division of the specification of the problem such as finding a minimum time solution by combining, e.g., low-lateral-acceleration solutions from the lower-level problem [23]. A bilevel interpretation of this is that the lower-level problem restricts the family of solutions the upper-level problem can use to compose an optimal solution.

B. Analysis of solution properties using bilevel arguments

From a practical point of view, the BOP consisting of (7) and (8) is in principle harder to solve than the standard formulation of the optimal control problem in (1). However, the formulation as a bilevel problem introduces possibilities to approximate the solution by sampling the solution to the lower-level mp-OCP as a function of its parameters. The result

of this sampling is that the solution to (8) is only computed for N_m predefined parameter combinations $(x_p^0, x_p^g, q_p) \in \mathcal{A}$, $p \in \{1,\dots,N_m\}$, where $\mathcal{A}$ is the user-defined set of combinations. These motion segments obtained by solving the mp-OCP for N_m parameters together constitute the motion primitive set $\mathcal{P}$ used in (3). An interpretation of this procedure is hence that $\mathcal{P}$ used in a lattice-based planner is a coarsely sampled parametric solution to the mp-OCP in (8) which can be used to represent the optimal solution of the lower-level problem when the upper-level problem is solved. The sampling introduces the well-known suboptimality of only being able to find solutions within the selected discretization [1]. However, this approximation makes it possible to solve a significant part of the bilevel problem offline and hence enables efficient graph-search methods to be used during online planning.

To be able to compute the motion primitives offline, the obstacle avoidance constraints in (8) are disregarded in the lower-level problem and instead handled during online planning in the upper-level problem. After this rearrangement, the BOP in (7) is equivalent to the lattice formulation (3). In the following theorem, it is shown that this rearrangement of constraints makes it in general impossible to obtain an optimal solution to a problem containing obstacles within the selected discretization since the lower-level problems are not required to satisfy the obstacle avoidance constraints.

Theorem 1: Let $\mathscr{P}_1$ denote the BOP

$$\begin{aligned} \underset{y,z}{\text{minimize}} \quad & F(y,z) \\ \text{subject to} \quad & z \in \arg\min_z \{F(y,z) : (y,z) \in \Omega\} \end{aligned} \tag{9}$$

with optimal objective function value $F(y_1^*, z_1^*)$. Furthermore, let $\mathscr{P}_2$ denote the BOP

$$\begin{aligned} \underset{y,z}{\text{minimize}} \quad & F(y,z) \\ \text{subject to} \quad & (y,z) \in \Omega \\ & z \in \arg\min_z \{F(y,z)\} \end{aligned} \tag{10}$$

with optimal objective function value $F(y_2^*, z_2^*)$. It then holds that $F(y_1^*, z_1^*) \leq F(y_2^*, z_2^*)$.

Proof 1: The feasible set of $\mathscr{P}_1$ is $Z_1 = \{(y,z) \mid z \in \arg\min_z \{F(y,z) : (y,z) \in \Omega\}\}$, and the feasible set of $\mathscr{P}_2$ is $Z_2 = \{(y,z) \mid (y,z) \in \Omega;\ z \in \arg\min_z \{F(y,z)\}\}$. Hence, any point in Z_2 is also in Z_1, i.e., $Z_2 \subseteq Z_1 \implies F(y_1^*, z_1^*) \leq F(y_2^*, z_2^*)$.

Definition 1: The active set $\mathcal{A}(y,z)$ at a feasible pair (y,z) of (9) consists of the inequality constraints that hold with equality [11], i.e.,

$$\mathcal{A}(y,z) = \{j \in \{1 \dots D\} \mid g_j(x,y) = 0\}.$$

Definition 2: Let (y^*, z^*) be an optimal solution to an optimization problem with KKT conditions satisfied with Lagrange multipliers λ^* associated to the inequality constraints in Ω. A constraint $g_j(y,z)$ in Ω is then said to be *strongly active* if $g_j(y^*, z^*) = 0$ and $\lambda_j^* > 0$ [11].

Corollary 1: Assume that the optimal solution (y_1^*, z_1^*) to $\mathscr{P}_1$ in (9) is unique. Then, if there exists a j such that $g_j(y_1^*, z_1^*)$ is strongly active in the lower-level problem, it holds that $F(y_1^*, z_1^*) < F(y_2^*, z_2^*)$ where (y_2^*, z_2^*) is the optimal solution to $\mathscr{P}_2$ in (10).

Proof 2: Since there exists at least one constraint which is strongly active at the lower level, it follows that $(y_1^*, z_1^*) \notin Z_2$, since $(y_1^*, \arg\min_z \{F(y_1^*, z)\}) \notin \Omega$. Hence, $(y_1^*, z_1^*) \in Z_1 \setminus Z_2$. Since (y_1^*, z_1^*) is the unique optimal solution to $\mathscr{P}_1$ over $Z_1 \supseteq Z_2$, it follows that $\nexists\, (y_2^*, z_2^*) \in Z_2 : F(y_2^*, z_2^*) \leq F(y_1^*, z_1^*)$. Hence, $F(y_1^*, z_1^*) < F(y_2^*, z_2^*)$.

An interpretation of Corollary 1 is that if the optimal solution to (7) is "strongly" in contact with the environment, then it is not in general possible to obtain an optimal solution using solutions to the lower-level problem (i.e., motion primitives) computed *without considering obstacles*. Note that these effects are beyond the fact that lower-level problems are sampled on a grid. The lower-level family of solutions is no longer optimal, instead the solutions need to adapt to the surrounding environment to become optimal, which is not a part of the standard lattice planning framework.

The approximate (but feasible) solution to (1) obtained by solving (3) using a lattice-based planner will inherently suffer from the suboptimality aspects shown in this section. One such example is shown in Fig. 3, where the car-like vehicle has to take an unnecessarily long route to avoid the obstacles because of the restricted search space. In the following section, it will be shown how to improve a suboptimal solution computed by a lattice-based planner by using direct methods for optimal control.

IV. Improvement using optimal control

In this section, we propose to use optimal control to improve the approximate solution computed by the lattice-based planner. By letting the system mode sequence $\{q_k\}_{k=1}^M$ be fixed to the solution from the lattice-based planner, the following OCP is obtained:

$$\begin{aligned} \underset{\{x_k(\cdot), u_k(\cdot),\, S_k\}_{k=0}^{M-1}}{\text{minimize}} \quad & \sum_{k=0}^{M-1} \int_0^{S_k} \ell\big(x_k(s), u_k(s), q_k\big)\mathrm{d}s \\ \text{subject to} \quad & x_0(0) = x_{\text{init}}, \quad x_{M-1}(S_{M-1}) = x_{\text{term}}, \\ & x_k'(s) = f_{q_k}(x_k(s), u_k(s)), \\ & x_{k+1}(0) = x_k(S_k), \\ & x_k(s) \in \mathcal{X}_{\text{free}}, \quad u_k(s) \in \mathcal{U}, \end{aligned} \tag{11}$$

where the optimization variables are the states $x_k(\cdot)$, control inputs $u_k(\cdot)$ and lengths S_k of the M phases. The difference compared to the optimal path planning problem (1) is that the combinatorial aspect of selecting the system mode sequence $\{q_k\}_{k=0}^{M-1}$ is already specified. However, since the length of the phases are optimization variables (with $\sum_k S_k = S_f$), it is possible that redundant phases introduced by the lattice-based planner are removed by selecting their lengths to zero. Furthermore, the second combinatorial aspect of selecting how to pass obstacles is implicitly encoded in the warm-start solution from the lattice-based planner. Finally, the difference in (11) compared to the lattice-based approximation in (3) is that the initial and terminal states for each phase are no longer restricted to the discretized state space $\mathcal{X}_d$, and the obstacles are explicitly considered when the motion in each phase is computed.

The problem in (11) is in the form of a standard multi-phase OCP, where the subsequent phases are connected using equality constraints. This problem can be solved using optimal control techniques, for example by applying a direct method to reformulate the problem as an NLP problem [12]. Today, there exist high-performing open-source NLP software such as IPOPT [30], WORHP [31], etc., that can be used to solve these types of problems. Common for such NLP solvers is that they aim at minimizing both the constraint violation and the objective function value [11]. Hence, for fast convergence to a local optimum, an initialization strategy should be selected that considers both the objective function and feasibility. In this work, the resolution-optimal solution $\{m_k\}_{k=1}^{M}$ from the lattice-based planner is used to initialize the NLP solver. It represents a path that is not only feasible, but also where each phase in the path (i.e., each motion primitive) has been computed by optimizing the *same* cost function $\ell(x,u,q)$ as in (11). Hence, the NLP solver is provided with a well-informed warm-start, which in general will decrease the time for the NLP solver to converge to a locally optimal solution [11]. Furthermore, when the same objective function is used both in (3) and (11), the NLP solver will often be initialized close to a good local minimum. Finally, a benefit of using a feasible initialization is that it is *always* guaranteed that a feasible solution exists that is homotopic with the provided initialization (at the very least the initialization itself), making it reliable to use online. Due to all these properties, the step of solving (11) is referred to as an improvement step in this work, which is somewhat in contrast to previous work where this step is commonly denoted "smoothing". Its primary aim is to improve the solution obtained from the lattice-based planner in terms of the objective function value.

Remark 1: Note that the improvement step, in contrast to only using the lattice-based planner, also can be used to enable path planning from and to initial and terminal states that are not within the specified state-space discretization $\mathcal{X}_d$. Here, the lattice-based planner can be used to find a path from and to the closest states in $\mathcal{X}_d$, and the improvement step can then adapt the path such that it starts at the initial state and reaches the terminal state exactly. However, in this case the warm-start cannot be guaranteed to be feasible.

After the improvement step is applied, a solution with lower objective function value compared to the solution from the lattice-based planner will in general be obtained since, relating back to Section III-B, the discretization constraints in the bilevel formulation are removed, and the paths are constructed while explicitly considering obstacles. One illustrative example of this is shown in Fig. 1 where the improved solution is noticably shorter compared to the suboptimal solution computed by the lattice-based planner.

A solution to the path planning problem is found using a preparation step offline and a two-step procedure online according to Algorithm 1. In the offline preparation step, the objective functions used in the motion primitive generation, the graph search in the lattice-based planner and the improvement step are specified. Furthermore, the system modes with associated vehicle models (used in the motion primitive generation and improvement step) are defined. Then, the motion primitive set $\mathcal{P}$ is computed by solving the N_m OCPs defined by the user without considering obstacles. For a detailed explanation of this step, the reader is referred to [20], [32].

Algorithm 1 Proposed path planning approach

1: **Offline**:
2: Specify $x(s)$, $\mathcal{X}$, $u(s)$, $\mathcal{U}$, $q \in \mathcal{Q}$, $f_q(x(s),u(s)) \in \mathcal{F}$ and $L_u(m)$, $\ell_l(x,u,q)$ and $\ell(x,u,q)$.
3: Choose $\mathcal{X}_d$ and select (x_p^0, x_p^g, q_p), $p = \{1 \dots N_m\}$
4: $\mathcal{P} \leftarrow$ solve N_m OCPs (8) disregarding $\mathcal{X}_{\text{obst}}$.
5: **Online**:
6: INPUT: x_{init}, x_{term}, $\mathcal{X}_{\text{obst}}$.
7: LATTICE-BASED PLANNER : $\{m_k, q_k\}_{k=0}^{M-1} \leftarrow$ Solve (3) from x_{init} to x_{term} with $\mathcal{P}$.
8: IMPROVEMENT : $\{x_k(\cdot), u_k(\cdot), S_k\}_{k=0}^{M-1} \leftarrow$ Solve (11) with $\{q_k\}_{k=0}^{M-1}$, warm-started with $\{m_k\}_{k=0}^{M-1}$.

The online procedure is initiated whenever a new path planning problem from x_{init} to x_{term} is requested be solved. In this step, a lattice-based planner is first used to solve the approximate path planning problem (3) using the precomputed motion primitive set $\mathcal{P}$ and the current description of the available free space $\mathcal{X}_{\text{free}}$. The lattice-based planner computes a resolution-optimal path, where the system mode is kept constant in each phase. This path is used as a well-informed warm-start to the final improvement step, where the multiphase OCP in (11) is solved to local optimality by improving the continuous aspects of the path computed by the lattice-based planner. The resulting path is then intended to be followed by a path-following controller, e.g., as in [5].

Remark 2: The online procedure can also be used for re-planning, e.g, if a new terminal state x_{term} is provided or if the environment $\mathcal{X}_{\text{obst}}$ has changed.

V. Numerical results

In this section, a proof-of-concept implementation of the proposed path planning approach is applied to two different vehicular systems: a car and a truck and trailer system. The lattice-based planner is implemented in C++ using A* graph search, where a precomputed free-space heuristic look-up table (HLUT) [33] is used as heuristic function to guide the search process. The HLUT is computed offline by solving path-planning problems in an obstacle-free environment from all discrete initial states $x_0 \in \mathcal{X}_d$ with a position at the origin to all final states $x_f \in \mathcal{X}_d$ with a position within a square centered around the origin with side length ρ. An appropriate size of the HLUT is both system and application dependent. In principle, a larger HLUT is required for less agile systems. In the simulations, the size of the HLUT is selected as $\rho = 40$ m for the car and $\rho = 80$ m for the truck and trailer system. For states that are not represented within the HLUT, the Euclidean distance is used as heuristic function. The motion-primitive computation and the improvement step are both implemented in Python using CasADi [34], where the warm-start friendly SQP method WORHP [31] is used as NLP solver. All simulations are performed on a laptop computer with an Intel Core i7-5600U processor. The total computation time for the offline steps, i.e., motion-primitive and HLUT computation, is roughly 20 minutes.

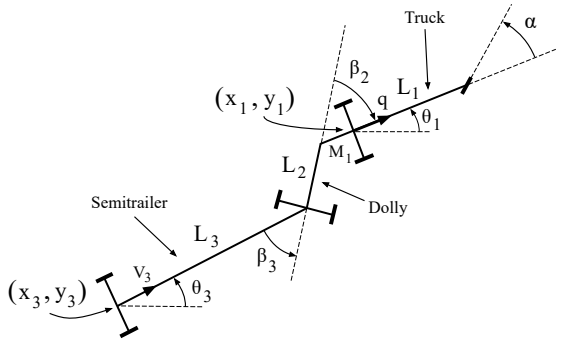

Fig. 4: A schematic illustration of the truck and trailer system that is used in the simulations.

A. Vehicle models

In the numerical experiments in this work, low-speed maneuvers are considered. As a consequence, kinematic vehicle models are used. The truck and trailer system is a general 2-trailer with a car-like truck [35]. This system is composed of three interconnected vehicle segments: a car-like truck, a dolly and a semitrailer, which is illustrated in Fig. 4. The state vector for this system is given by $x(s) = (\bar{x}(s), \alpha(s), \omega(s))$ where $\bar{x}(s) = (x_3(s), y_3(s), \theta_3(s), \beta_3(s), \beta_2(s))$. Here, (x_3, y_3) is the center of the axle of the semitrailer, θ_3 is the orientation of the semitrailer, β_3 is the joint angle between the semitrailer and the dolly, β_2 is the joint angle between the dolly and the car-like truck. The constraints imposed on the states and control signal are given by $|\alpha(s)| \leq \pi/4$, $|\omega(s)| \leq 0.5$ and $|u_\omega(s)| \leq 40$ (equal to the ones used in [5]). The model of this system can be compactly represented as (see [5] for details):

$$\begin{aligned} \bar{x}'(s) &= q f_t(\bar{x}(s), \alpha(s)), \\ \alpha'(s) &= \omega(s), \quad \omega'(s) = u_\omega(s), \end{aligned} \tag{12}$$

where $q \in \{1, -1\}$ represents the discrete decision variable which corresponds to the direction of motion. The vehicle's geometry (illustrated in Fig. 4) used in this section coincides with what is used in [5]. The cost function used for the truck and trailer system is given by

$$\ell_t(x, u_\omega, q) = \begin{cases} 1 + \gamma(\alpha^2 + 10\omega^2 + u_\omega^2), & q = 1, \\ 1 + \gamma(\beta_3^2 + \beta_2^2 + \alpha^2 + 10\omega^2 + u_\omega^2), & q = -1, \end{cases} \tag{13}$$

where the variable γ represents the trade-off between path length and smoothness of the solution. Furthermore, quadratic penalties for (large) joint angles β_3 and β_2 are added to the cost function for paths in backward motion to avoid so-called jack-knife states.

The model of the car is given in Example 1, where $L_1 = 2.9$ m is used as wheel-base. The car's steering angle α and its derivatives, ω and u_ω, are subject to the same constraints as in the truck and trailer case. The cost function used for the car is given by

$$\ell_c(x, u_\omega, q) = 1 + \gamma(\alpha^2 + 10\omega^2 + u_\omega^2). \tag{14}$$

Unless stated otherwise, $\gamma = 1$ is used in the cost functions in (13) and (14) to define the objective functions in the offline motion-primitive computation step (related to solving N_m problems in the form of (8)), the lattice-based planner in (3) and the improvement step in (11).

TABLE I: A description of the different motion-primitive sets used. $|\Theta|$ defines the number of heading discretization points, $\Delta_\theta^{\max}$ defines which neighboring headings to connect (from 1 to $\Delta_\theta^{\max}$) and n_{par} defines the number of parallel maneuvers (per heading). Finally, n_{prim} defines the resulting total number of motion primitives.

$\mathcal{P}$	$\lvert\Theta\rvert$	$\Delta_\theta^{\max}$	n_{par}	n_{prim}
$\mathcal{P}_{\text{geo}}$	16	2	N/A	240
$\mathcal{P}_{\text{kin}}$	16	4	3	480
$\mathcal{P}_{\text{com}}$	16	4	3	480

B. State lattice construction

To be able to compare the proposed method with other state-of-the-art initialization strategies, three different motion-primitive sets for each vehicle are used by the lattice-based planner. The sets are computed by using either simplified or complete vehicle models. The first motion-primitive sets $\mathcal{P}_{\text{com}}$ use the complete vehicle models. The second sets $\mathcal{P}_{\text{kin}}$ do not include ω and u_ω, making the steering angle α considered as control signal (i.e., purely kinematic models), which is similar to the initialization strategy used in [3]. The third sets $\mathcal{P}_{\text{geo}}$ are computed by completely neglecting the nonlinear system model, further referred to as a geometric model, where instead linear interpolation is used between the initial and final states for each motion primitive.

Before computing the motion-primitive sets, the state space of the vehicles need to be discretized. The positions, (x_1, y_1) for the car and (x_3, y_3) for the semitrailer, are discretized onto a uniform grid with resolution $r = 1$ m and the orientations $\theta_1 \in \Theta$ and $\theta_3 \in \Theta$ are irregularly discretized as proposed in [20]. The discretization of the steering angle α is only applicable for the complete models. For simplicity, it is here constrained to zero and its rate ω is also constrained to zero to ensure that α is continuously differentiable, even when motion segments are combined online [23]. For the truck and trailer system, the joint angles β_3 and β_2 are also constrained to zero at each discretized state in the state lattice. Note however that on the path between two discretized states, the systems can take any feasible vehicle configuration.

The motion-primitive sets are automatically computed using the approach described in [32], where the sets are composed of heading changes and parallel maneuvers according to Table I. These maneuvers are optimized using the cost functions defined in (13) and (14). For the simplified vehicle models, the neglected states are disregarded in the cost functions. For a more detailed description of the state-lattice construction, the reader is referred to [32].

C. Simulation results

For the car model, a parallel parking scenario (Fig. 5) is used. For the truck and trailer system, a loading-site area (Fig. 6) and a parking scenario (Fig. 7) are considered. The geometry of obstacles and vehicles are represented by bounding circles [1]. The area of the car is described by three circles, while the truck is described by one circle and the trailer by two circles. This choice of obstacle representation can be used in all steps used in the proposed framework since the constraints can be described by smooth functions. An alternative object representation that is compatible with the

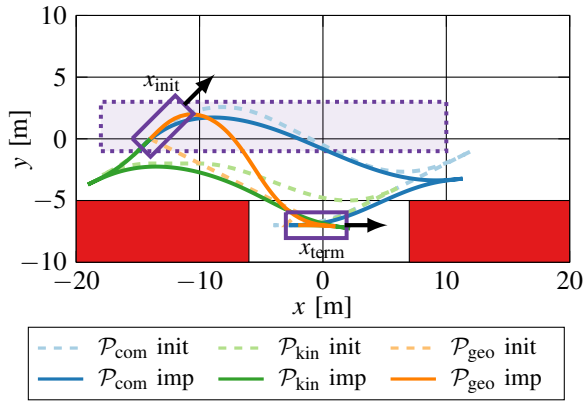

Fig. 5: Parallel parking scenario solved from several initial states with $\theta_1^i = \{0, \pi/4\}$ (indicated by area within the dotted lines). Solutions from one problem are illustrated for the three initialization strategies (using the motion-primitive sets described in Table I) with corresponding solutions from the improvement step.

TABLE II: Parallel parking scenario for the car-like vehicle (Fig. 5, 150 problems). $\mathcal{P}$ is the motion-primitive set used in the lattice-based planner. $\bar{t}_P$ is the average time for the lattice-based planner to find a solution. r_{imp} and $\bar{t}_{\text{imp}}$ are the success rate and average time for the improvement step to converge. $\bar{t}_{\text{tot}}$ is the average total time. Finally, $\bar{J}_{\text{P}}$ and $\bar{J}_{\text{imp}}$ is the average objective function value for the solutions from the lattice-based planner and improvement step, respectively.

$\mathcal{P}$	$\bar{t}_P$ [s]	$\bar{t}_{\text{imp}}$ [s]	$\bar{t}_{\text{tot}}$ [s]	r_{imp}	$\bar{J}_P$	$\bar{J}_{\text{imp}}$
$\mathcal{P}_{\text{geo}}$	0.0011	1.12	1.12	62 %	N/A	30.8
$\mathcal{P}_{\text{kin}}$	0.025	1.03	1.06	90.7 %	N/A	28.7
$\mathcal{P}_{\text{com}}$	0.014	0.88	0.894	100 %	35.7	27.5

approach presented in this work is proposed in [16], where vehicles and obstacles can be represented by convex sets.

The path planning problems are first solved by the lattice-based planner, using the three different motion-primitive sets described in Table I. Thereafter, the obtained solutions are used to initialize the improvement step. For the initializations based on the simplified models $\mathcal{P}_{\text{geo}}$ and $\mathcal{P}_{\text{kin}}$, all states that are not represented are initialized to zero.

For the car scenario in Fig. 5, the results in Table II show that the lattice-based planner achieves the lowest computation times when the geometric model is used, compared to the kinematic and the complete model. However, using this simple initialization strategy results in a decreased reliability (only 62 %) and the total average computation time $\bar{t}_{\text{tot}}$ becomes higher than the two other cases due to a more computationally demanding improvement step. The kinematic initialization performs better than the geometric in terms of reliability, but still fails to converge to a solution in almost 10 % of the simulations. When the complete model is used in the lattice-based planner, the computation time for the improvement step is significantly reduced compared to when the simpler initialization strategies are used. In particular, the total computation time including the lattice-based planner is as much as halved and the success rate is 100 % in all experiments. Furthermore, the mean objective function value $\bar{J}_{\text{opt}}$ decreases significantly compared to the solution from the lattice-based planner $\bar{J}_P$. For the two simpler initialization strategies, no comparable objective function values from the lattice-based planner exist

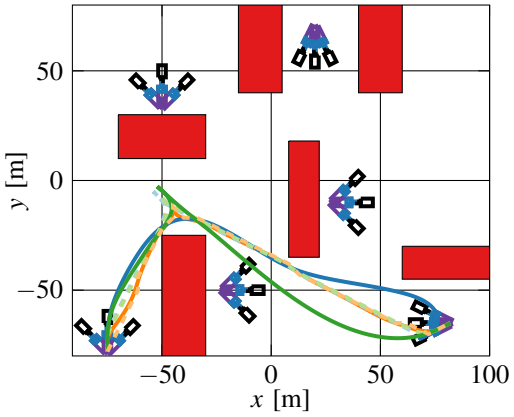

Fig. 6: Loading-site scenario for the truck and trailer system, solved from and to the states illustrated in the figure. See Fig. 5 for a further description of the content.

TABLE III: Loading site scenario for the truck and trailer system (Fig. 6, 270 problems). See Table II for a description of the variables.

$\mathcal{P}$	$\bar{t}_P$ [s]	$\bar{t}_{\text{imp}}$ [s]	$\bar{t}_{\text{tot}}$ [s]	r_{imp}	$\bar{J}_P$	$\bar{J}_{\text{imp}}$
$\mathcal{P}_{\text{geo}}$	0.69	18.4	19.1	43.7 %	N/A	236
$\mathcal{P}_{\text{kin}}$	6.5	15.0	21.5	98.9 %	N/A	164
$\mathcal{P}_{\text{com}}$	5.35	9.45	14.8	100 %	184	164

since they are infeasible to the actual path-planning problem.

The results for the truck and trailer system (Fig. 6-7) are summarized in Table III-IV. In these simulations, using a feasible initialization (as proposed in this work) has an even larger impact on the time spent in the improvement step. The reason why such a large computational performance gain is obtained in these more advanced scenarios is mainly due to the complicated nonlinear system model, which also affects the reliability when using a geometric initialization strategy where the success rate is less than 50 % in the loading-site scenario. It can also be noted that the computational effort required to solve the problems in this scenario is significantly higher compared to the parking scenarios. This is due to an increased dimensionality of the resulting NLP caused by longer plans. The online planning time can be reduced by, e.g., performing the improvement step in a receding-horizon fashion. However, this is out-of-scope in this work and left as future work. Finally, the reliability for the kinematic initialization is higher compared to the car scenario. However, the best overall performance in terms of total computation time, numerical reliability and objective function value is still obtained by using the same model in both steps, which is the approach introduced in this work.

In Fig. 8, the steering angles obtained along the paths from the lattice-based planner and the improvement step for one of the problems in the parking scenario (Fig. 7) are compared. The results show that unnecessary steering effort is removed by the improvement step, especially in backward motion since large joint angles are penalized to avoid jack-knife configurations when the plan is executed. Furthermore, a redundant initial backward phase in the solution from the lattice-based planner, caused by discretization artifacts, is removed in the improvement step (as discussed in Section IV).

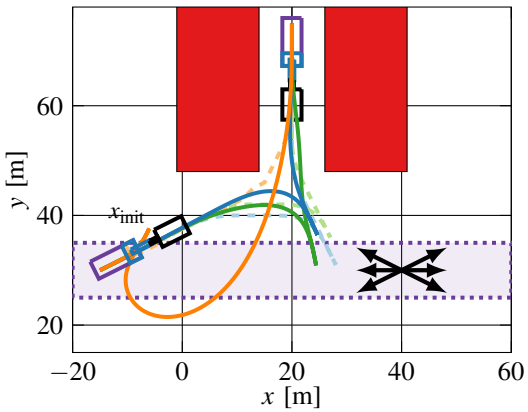

Fig. 7: Parking scenario for the truck and trailer system solved from several initial states within the area indicated by the dotted lines. The black arrows represent initial headings used. See Fig. 5 for a further description of the content.

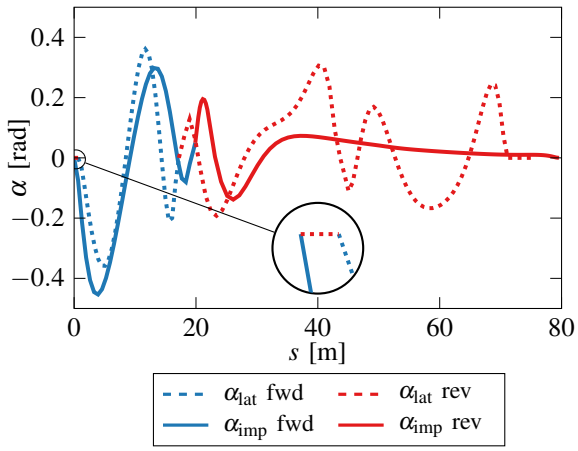

Fig. 8: $\alpha(s)$ obtained from the lattice-based planner (dashed) and the improvement step (solid) for one of the solutions to the truck and trailer parking scenario in Fig. 7. Blue and red colors represent forward and backward motion, respectively. Note that the improvement step removes a redundant initial phase of backward motion (highlighted by the magnified part).

In Table V, we evaluate the impact of using the complete vehicle model and the same or different objective functions at the three steps; the motion-primitive computation, the graph search in the lattice-based planner and the improvement step. The impact is investigated by varying the value of γ in (13) used in the graph search (γ_u) and the motion-primitive computation (γ_l), while it is kept constant in the improvement step (γ_i). The results on row one in Table V ($\gamma_u = \gamma_l = \gamma_i = 10$) represent the proposed approach where the same cost function is used to define the objective functions in all steps. When shortest path is used as objective function in the graph search ($\gamma_u = 0$, row two in Table V), the average cost for a path after the improvement step is increased by roughly 10 % compared to the baseline, due to that the improvement step converges to a worse local minimum. However, the total computation time decreases, mainly as a result of a faster graph search. The computation time for the improvement step is similar to using $\gamma_u = 10$, which is reasonable since each motion primitive in the warm-start is optimized using the same objective function as in the improvement step. When also the motion primitives are

TABLE IV: Parking scenario for the truck and trailer system (Fig. 7, 306 problems). See Table II for a description of the variables.

$\mathcal{P}$	$\bar{t}_P$ [s]	$\bar{t}_{\text{imp}}$ [s]	$\bar{t}_{\text{tot}}$ [s]	r_{imp}	$\bar{J}_P$	$\bar{J}_{\text{imp}}$
$\mathcal{P}_{\text{geo}}$	0.016	6.17	6.18	86.6 %	N/A	102
$\mathcal{P}_{\text{kin}}$	0.016	2.72	2.74	99.0 %	N/A	82.7
$\mathcal{P}_{\text{com}}$	0.020	2.08	2.10	100 %	96.2	82.3

TABLE V: Loading site scenario for the truck and trailer system (Fig. 6, 270 problems). γ_u, γ_l and γ_i are the values of γ in (13) used in the graph search, the motion-primitive computation and improvement step, respectively. See Table II for a description of the other variables.

$\mathcal{P}$	γ_u	γ_l	γ_i	$\bar{t}_P$	$\bar{t}_{\text{imp}}$	$\bar{t}_{\text{tot}}$	r_{imp}	$\bar{J}_{\text{imp}}$
$\mathcal{P}_{\text{com}}$	10	10	10	3.2	6.2	9.4	100 %	190
$\mathcal{P}_{\text{com}}$	0	10	10	2.2	5.9	8.1	100 %	208
$\mathcal{P}_{\text{com}}$	0	0	10	1.9	12.6	14.5	100 %	212

generated using shortest path as objective function ($\gamma_u = \gamma_l = 0$, row 3 in Table V), not only the average solution cost increases, but also the convergence time for the improvement step. The reason is that each phase in the initialization is far from a local minimum in terms of the objective function used in the improvement step. This clearly illustrates the importance of using the same objective function in the motion-primitive computation and improvement step for fast convergence in the latter step, which is what is proposed in this work.

VI. Conclusions and Future Work

A unified optimization-based path planning approach to compute high-quality locally optimal solutions to path planning problems is proposed. The first step of the proposed approach consists of using a lattice-based path planner to find a resolution-optimal path using a discretized search space. This path is then used as a well-informed warm-start in a second improvement step where an optimal control-based method is used to compute a locally optimal solution to the path planning problem. The use of these steps has been motivated by showing that a lattice-based path planner can be cast and analyzed as a bilevel optimization problem. To tightly couple the two steps, the lattice-based planner uses a system model and objective function that are chosen to coincide with those used in the second improvement step. This new, more carefully selected combination of a path planner and optimal control makes, in a structured way, benefit of the former method's ability to solve combinatorial parts of the problem and the latter method's ability to obtain locally optimal solutions not restricted to a discretized search space. The value of this new tight combination is thoroughly investigated with successful results in several practically relevant path planning problems where it is shown to outperform state-of-the-art initialization strategies in terms of computation time, numerical reliability, and objective function value.

Future work includes to decrease the online planning time by, e.g., optimizing the proof-of-concept implementation, performing the improvement step in a receding-horizon fashion and/or to use ideas from fast MPC such as [36]. Another extension is to investigate how to apply the approach in high-speed and dynamic scenarios. Finally, another interesting direction for future work is to apply the approach to systems with more distinct system modes, such as morphable drones [25].

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proceedings of the 57th IEEE Conference on Decision and Control*, 2018, pp. 4327–4332.

[4] P. F. Lima, M. Nilsson, M. Trincavelli, J. Mårtensson, and B. Wahlberg, "Spatial model predictive control for smooth and accurate steering of an autonomous truck," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 4, pp. 238–250, 2017.

[5] O. Ljungqvist *et al.*, "A path planning and path-following control framework for a general 2-trailer with a car-like tractor," *Journal of field robotics*, vol. 36, no. 8, pp. 1345–1377, 2019.

[6] A. Piazzi, C. G. L. Bianco, and M. Romano, "η^3-splines for the smooth path generation of wheeled mobile robots," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1089–1095, 2007.

[7] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 561–568, 2010.

[8] R. Oliveira, M. Cirillo, and B. Wahlberg, "Combining lattice-based planning and path optimization in autonomous heavy duty vehicle applications," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 2090–2097.

[9] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness, motion planning and trailer systems," in *Proceedings of 32nd IEEE Conference on Decision and Control*, vol. 3, 1993, pp. 2700–2705.

[10] I. E. Grossmann, "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and engineering*, vol. 3, no. 3, pp. 227–252, 2002.

[11] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[12] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sci.*, vol. 135, no. 1, pp. 497–528, 2009.

[13] K. Bergman and D. Axehill, "Combining homotopy methods and numerical optimal control to solve motion planning problems," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 347–354.

[14] L. Campos-Macías *et al.*, "A hybrid method for online trajectory planning of mobile robots in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 935–942, 2017.

[15] H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. J. Lilienthal, "Fast, continuous state path smoothing to improve navigation accuracy," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 662–669.

[16] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, 2020.

[17] S. Stoneman and R. Lampariello, "Embedding nonlinear optimization in RRT* for optimal kinodynamic planning," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014, pp. 3737–3744.

[18] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[19] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5041–5047.

[20] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[22] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems 16*, 2004, pp. 767–774.

[23] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer, "Lattice-based motion planning for a general 2-trailer system," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[24] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 4, 1999, pp. 3972–3977.

[25] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza, "The foldable drone: A morphing quadrotor that can squeeze and fly," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 209–216, 2019.

[26] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2172–2179.

[27] ——, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2611–2616.

[28] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Ann. of operations res.*, vol. 153, no. 1, pp. 235–256, 2007.

[29] N. P. Faísca, V. Dua, B. Rustem, P. M. Saraiva, and E. N. Pistikopoulos, "Parametric global optimisation for bilevel programming," *Journal of Global Optimization*, vol. 38, no. 4, pp. 609–623, 2007.

[30] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. programming*, vol. 106, no. 1, pp. 25–57, 2006.

[31] C. Büskens and D. Wassel, "The ESA NLP solver WORHP," in *Modeling and Optimization in Space Engineering*, G. Fasano and J. D. Pintér, Eds. Springer New York, 2013, vol. 73, pp. 85–110.

[32] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved optimization of motion primitives for motion planning in state lattices," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, June 2019.

[33] R. A. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3375–3380.

[34] J. A. E. Andersson *et al.*, "CasADi – A software framework for nonlinear optimization and optimal control," *Math. Progam. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[35] C. Altafini, A. Speranzon, and K.-H. Johansson, "Hybrid control of a truck and trailer vehicle," in *Hybrid Systems: Computation and Control*. Springer, 2002, pp. 21–34.

[36] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

Kristoffer Bergman is a PhD student in the Division of Automatic Control, Department of Electrical Engineering, Linköping University. He received his M.Sc. degree in Electrical Engineering in 2014 and his Lic. Eng. degree in Automatic Control in 2019, both from Linköping University. He is currently also working as a systems engineer at Saab Dynamics. His research interests include motion planning, optimization and optimal control.

Oskar Ljungqvist is a PhD student in the Division of Automatic Control, Department of Electrical Engineering, Linköping University. He received his M.Sc. degree in Applied Physics and Electrical Engineering in 2015 and his Lic. Eng. degree in Automatic Control in 2019, both from Linköping University. His research interests include motion planning, state estimation and feedback control of tractor-trailer vehicles.

Daniel Axehill received his M.Sc. degree in Applied Physics and Electrical Engineering in 2003. Furthermore, he received the degree of Lic. Eng. in Automatic Control in 2005 and the Ph.D. degree in Automatic Control in 2008. All three degrees are from Linköping University, Linköping, in Sweden. In year 2006 he spent three months at UCLA in Los Angeles. From January 2009 and until November 2010 he worked as a post-doc at the Automatic Control Laboratory at ETH Zurich. He is currently employed as an Associate Professor at the Division of Automatic Control at Linköping University. His research interests are related to optimization, optimal control, motion planning, hybrid systems, and applications of control.