Equivariant Deep Learning of Mixed-Integer Optimal Control Solutions for Vehicle Decision Making and Motion Planning

Rudolf Reiter, Rien Quirynen, Moritz Diehl, and Stefano Di Cairano, *Senior Member, IEEE*

***Abstract*— Mixed-integer quadratic programs (MIQPs) are a versatile way of formulating vehicle decision making (DM) and motion planning problems, where the prediction model is a hybrid dynamical system that involves both discrete and continuous decision variables. However, even the most advanced MIQP solvers can hardly account for the challenging requirements of automotive-embedded platforms. Thus, we use machine learning to simplify and hence speed up optimization. Our work builds on recent ideas for solving MIQPs in real time by training a neural network (NN) to predict the optimal values of integer variables and solving the remaining problem by online quadratic programming. Specifically, we propose a recurrent permutation equivariant deep set (REDS) that is particularly suited for imitating MIQPs that involve many obstacles, which is often the major source of computational burden in motion planning problems. Our framework comprises also a feasibility projector (FP) that corrects infeasible predictions of integer variables and considerably increases the likelihood of computing a collision-free trajectory. We evaluate the performance, safety, and real-time feasibility of DM for autonomous driving using the proposed approach on realistic multilane traffic scenarios with interactive agents in SUMO simulations.**

***Index Terms*— Geometric deep learning, mixed-integer optimization, motion planning.**

I. Introduction

DECISION making (DM) and motion planning for automated driving (AD) is challenging due to several reasons [1]. First, in general, even formulations of deterministic, 2-D motion planning problems are PSPACE-hard [2], [3]. Second, (semi-)autonomous vehicles operate in highly dynamic environments, thus requiring a relatively high-control update rate. Finally, there is always uncertainty that stems from model mismatch, inaccurate measurements as well as other drivers' unknown intentions. The complexity of motion planning and DM for AD and its real-time requirements in resource-limited automotive platforms [4] requires the implementation of a multilayer guidance and control architecture [1], [5].

Based on a route given by a navigation system, a DM module decides what maneuvers to perform, such as lane changing, stopping, waiting, and intersection crossing. Given the outcome of such decisions, a motion planning system generates a trajectory to execute the maneuvers, and a vehicle control system computes the input signals to track it. Recent work [6] presented a mixed-integer programming-based vehicle decision maker (MIP-DM), which simultaneously performs maneuver selection and trajectory generation by solving a mixed-integer quadratic program (MIQP) at each time instant. In this article, we present an algorithm to implement MIP-DM based on supervised learning and sequential quadratic program (SQP), to compute a collision-free and close-to-optimal solution with a considerably reduced online computation time compared with advanced MIQP solvers.

The presented approach consists of an *offline* supervised learning procedure and an *online* evaluation step that includes a feasibility projector (FP). In the *offline* procedure, expert data are collected by computing the exact solutions of the MIQP for a large number of samples from a distribution of parameter values in the MIP-DM, including, for example, states of the autonomous vehicle, its surrounding traffic environment, and speed limits. Along the paradigm of [7] and [8], a neural network (NN) is trained with the collected expert data to predict the binary variables that occur within the MIQP, which are the main source of the computational complexity of MIQPs. A novel NN architecture, referred to as recurrent EDS (REDS), is proposed that exploits key structural domain properties, such as permutation equivariance related to obstacles and recurrence of the time series.

In the *online* evaluation step, the NN predicts the optimal values of the binary variables in the MIQP. After fixing these, the resulting problem becomes a convex quadratic program (QP) that can be solved efficiently. To account for potentially wrong predictions, the QP is formulated with slack variables (soft-QP). The soft-QP solution is forwarded to a FP to correct any infeasibilities, implemented by a nonlinear program (NLP) with smooth, but concave obstacle constraints. Such convex–concave NLPs can be solved efficiently using an

Manuscript received 28 October 2023; revised 12 April 2024; accepted 29 April 2024. Date of publication 6 June 2024; date of current version 26 June 2025. The work of Rudolf Reiter and Moritz Diehl was supported in part by the European Union (EU) under Grant ELO-X 953348, in part by the Deutsche Forschungsgemeinschaft (DFG) via Research Unit for 2401 under Project 424107692 and Project 525018088, and in part by the Bundesministerium für Wirtschaft und Klimaschutz (BMWK) under Grant 03EI4057A and Grant 03EN3054B. Recommended by Associate Editor Y. Chen. *(Corresponding author: Rudolf Reiter.)*

Rudolf Reiter is with the Department of Microsystems Engineering (IMTEK), University of Freiburg, 79110 Freiburg, Germany (e-mail: rudolf.reiter@imtek.uni-freiburg.com).

Rien Quirynen was with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139 USA (e-mail: rien.quirynen@gmail.com).

Moritz Diehl is with the Department of Microsystems Engineering (IMTEK) and the Department of Mathematics, University of Freiburg, 79110 Freiburg, Germany (e-mail: moritz.diehl@imtek.uni-freiburg.com).

Stefano Di Cairano is with the Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: dicairano@merl.com).

Digital Object Identifier 10.1109/TCST.2024.3400571

1063-6536 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

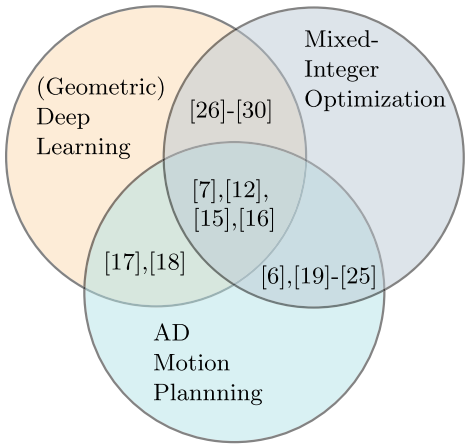

Fig. 1. Categorical overview of related work. The research domain of the proposed approach is located at the intersection of three areas.

SQP algorithm [9], [10]. To further increase the likelihood of finding a feasible and possibly optimal solution, an ensemble of NNs is trained and evaluated, and the "best" solution is selected at each time step.

The overall performance of the proposed method is compared against the advanced MIQP solver `gurobi` [11], and alternative NN architectures [12] and evaluated in high-fidelity closed-loop simulations using `SUMO` [13] and `CommonRoad` [14].

A. Related Works

This work lies at the intersection of three research areas, i.e., geometric deep learning, mixed-integer programming (MIP), and motion planning for AD (see Fig. 1). Motion planning problems are solved by algorithms that can handle combinatorial complexity and dynamic feasibility under real-time computational limits [15], [16].

Motion primitives are appealing due to their simplicity and alignment with the road geometry [17]. However, the motion plans are usually suboptimal or conservative.

In several works, the graph search is performed on a discretized state space [1]. Probabilistic road maps [18] and rapidly exploring random trees [19] are common in highway motion planning. Nonetheless, they suffer from the curse of dimensionality, a poor connectivity graph, and no repeatability [15]. By using heuristics, A^* [20] aims to avoid the problem of high-dimensional discretization. However, choosing an appropriate heuristic is challenging and graph generation in each iteration is time-consuming [1].

For some conservative simplifications related to highway driving, the state space can be decomposed into spatiotemporal driving corridors [21], [22], [23]. Such nonconvex regions can be further decomposed into convex cells [24] or used in a sampling-based planner [23]. For these cases, the performance is limited due to the required over-approximations.

Derivative-based numerical optimization methods can successfully solve problems in high-dimensional state spaces in real-time [25]. However, these approaches are often restricted to convex problem structures or sufficiently good initial guesses. While highway motion planning and DM are highly nonconvex, by introducing integer variables, the problem can be formulated as an MIQP [6], [23], [26], [27], [28], [29], [30], [31].

A common problem is the significant computation time of MIQP solvers. Authors [6], [27], [30], [32] mitigate the problem by either scaling down the problem size, neglecting real-time requirements [28] or accepting the high computation time for non-real-time simulations while leaving real-time feasibility open for the future work [26], [29], [31].

Several recent works use deep learning to accelerate MIQP solutions. One strategy is to improve algorithmic components within an online MIQP solver [33], [34], [35]. Masti and Bemporad [33] use deep learning to warm-start MIQP solvers, which, however, cannot lower the worst case computation time. Nair et al. [34] and Khalil et al. [35] propose a learning strategy to guide a branch-and-bound (BB) algorithm. For solving mixed-integer linear programs (MILPs), Russo et al. [36] use NNs for a custom solver to achieve similar computation times as commercial solvers, which, however, may still be large. Another strategy is based on supervised or imitation learning, using MIQP solutions as expert data to train function approximators offline [37], [38], [39]. Bertsimas and Stellato [37] show how the classification of binary variables in MIQPs can produce high-quality solutions with a low computation time. Recent work [12] extends supervised learning for MIQP-based motion planning [7] using a recurrent NN (RNN) and presolve techniques to increase the likelihood of predicting a feasible solution. Due to the RNN, the results in [12] scale well with the horizon length, but not yet with the number of obstacles, as shown later.

None of the works that use deep learning to accelerate MIQP-based motion planning consider or leverage geometric deep learning, particularly, permutation equivariance or invariance, to decrease the network size and increase the performance. As one consequence, they do not allow variable input and variable output dimensions, e.g., corresponding to a variable number of obstacles. However, some recent works successfully use geometric deep learning for related tasks in AD, e.g., the prediction of other road participants [40], [41], or within reinforcement learning (RL) [42].

Other techniques for decision making (DM) exclusively use NNs, without solving optimization problems online, e.g., using deep RL [43]. These approaches usually require more data, are less interpretable, and may lack safety without additional safety layers, due to the approximate nature of the NN output [15].

B. Contributions

Our main contribution is an REDS for the NN predicting integer variables of MIQPs, which is particularly suited for learning time-series and obstacle-related binary variables in motion planning problems. In particular, the recurrent permutation equivariant deep set (REDS) enables the NN to predict binary variables for collision avoidance concerning a varying number of obstacles, and the predicted solution is the same regardless of the order in which the obstacles are

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

provided. Our proposed framework includes an ensemble of NNs in combination with an FP that increases the likelihood of computing a collision-free trajectory. Compared with the state of the art, we show that our method improves the prediction accuracy, adds permutation equivariance, allows for a variable number of obstacles and horizon length, and generalizes well to unseen data, such as several obstacles not present in the training data. As a final contribution, we demonstrate the performance of a novel integrated planning system, which is further referred to as REDS planner. The REDS planner uses an ensemble of REDSs, a selection of the best soft-QP solution, and the FP for real-time vehicle DM and motion planning. We present closed-loop simulation results with reference tracking by a nonlinear model predictive controller (NMPC) for realistic traffic scenarios, using interactive agents in `SUMO` [13], and a high-fidelity vehicle model and the problem setup provided by `CommonRoad` [14].

C. Preliminaries and Notation

The notation $I(n) = \{z \in \mathbb{N} | 0 \leq z \leq n\}$ is used for index sets and $\mathbb{B} = \{0, 1\}$. Throughout this article, the attributes of *equivariance (invariance)* exclusively refer to *permutation equivariance (invariance)* with the following definition.

Definition 1: Let $f(\zeta^{\text{us}}):\mathbb{X}^M \to \mathbb{Y}$ be a function on a set of variables $\zeta^{\text{us}} = \{\zeta_1^{\text{us}}, \ldots, \zeta_M^{\text{us}}\} \in \mathbb{X}^M$ and let $\mathcal{G}$ be the permutation group on $\{1, \ldots, M\}$. The function f is *permutation invariant*, if $f(g \cdot \zeta^{\text{us}}) = f(\zeta^{\text{us}})$ for all $g \in \mathcal{G}, \zeta^{\text{us}} \in \mathbb{X}^M$.

Definition 2: Let $f(\zeta^{\text{eq}}):\mathbb{X}^N \to \mathbb{Y}^N$ be a function on a set of variables $\zeta^{\text{eq}} = \{\zeta_1^{\text{eq}}, \ldots, \zeta_N^{\text{eq}}\} \in \mathbb{X}^N$ and let $\mathcal{G}$ be the permutation group on $\{1, \ldots, N\}$. The function f is *permutation equivariant*, if $f(g \cdot \zeta^{\text{eq}}) = g \cdot f(\zeta^{\text{eq}})$ for all $g \in \mathcal{G}, \zeta^{\text{eq}} \in \mathbb{X}^N$.

Features that are modeled without structural symmetries are referred to as *unstructured features*. Lower and upper bounds on decision variables x are denoted by $\underline{x}$ and $\overline{x}$, respectively. The all-one vector $[1, \ldots, 1]^\top$ of size n is $\mathbf{1}_n^\top$. The notation $f(x; y)$ in the context of optimization problems indicates that function $f(\cdot)$ depends on decision variables x and parameters y. Lower case letters $x \in \mathbb{R}^n$ refer to scalars or vectors of size n, and their upper case version $X \in \mathbb{R}^{n \times N}$ refer to the matrix associated with a sequence of those vectors along a time horizon N. *Obstacles* normally refer to surrounding vehicles.

II. Problem Setup and Formulation

In this work, an autonomous vehicle shall drive *safely* along a multilane road, while obeying the traffic rules. We consider DM based on MIP that determines the driving action and a reference trajectory for the vehicle control to follow. The definition of *safety* can be ambiguous [44]. We use the term *safe* to refer to satisfying hard collision avoidance constraints concerning known obstacles' trajectories. Our problem setup relies on the following assumptions.

Assumption 1: There exists a prediction time window along which the following are known.

1) The predicted position and orientation for each of the obstacles in a neighborhood of the ego vehicle up to a sufficient precision,
2) The high-definition map information, including center lines, road curvature, lane widths, and speed limits.

Assumption 1 requires the vehicle to be equipped with on-board sensors and perception systems [45], an obstacle prediction module [1], [5], the high-definition map database, and either on-board or obtained through vehicle-to-infrastructure (V2I) communication [46]. The effect of prediction inaccuracies and uncertainty on vehicle safety is outside the scope of this article, but relevant work can be found in [1] and [5] and references therein.

Assumption 2: We assume a localization with centimeter-level accuracy at each sampling time.

Assumption 2 is possible thanks to recent advances in GNSS that achieves centimeter-level accuracy at a limited cost [47], [48].

We propose a DM that satisfies the following requirements.

Requirement 1: The DM must plan a sequence of maneuvers, possibly including one or multiple lane changes, and a corresponding collision-free trajectory to make progress along the vehicle's future route with a desired nominal velocity.

Requirement 2: The DM yields kinematically feasible trajectories, satisfying vehicle limitations and traffic rules (e.g., speed limits).

Requirement 3: The DM must be agnostic to permutations of surrounding vehicles, i.e., the plan must be consistent, irrespective of the order the vehicles are processed (see Definition 2).

Requirement 4: The DM must hold a worst-case computation time lower than the real-time planning period t_p, with a target value of $t_p \leq 0.2$ s.

Requirement 5: At any time, the DM must satisfy all requirements, regardless of the number of surrounding vehicles.

Based on Assumption 1, Requirements 1–3 can be met by the MIP-DM from [6]. However, the crucial Requirements 4 and 5 may be difficult to meet by the MIP-DM, when executing on embedded control hardware with limited computational resources and memory [4]. The main focus of this work is approximating the MIP-DM with a suitable framework that satisfies all Requirements 1–5. Notably, Requirement 3 is trivially true for the MIP-DM [6]. However, this is not the case for NN-based planners unless the architecture is invariant to permutations [49].

A. Nominal and Learning-Based Architecture

A hierarchical control architecture (see Fig. 2) of a DM is proposed, followed by a reference tracking NMPC. Within the architecture of Fig. 2, the expert MIP-DM is used as a benchmark. We refer to the module aiming to imitate the expert MIP-DM as the REDS planner. It comprises an ensemble of NNs that predict the binary variables of the MIQP, as used in the expert MIP-DM. For each NN, a soft-QP is solved with the binary variables fixed to the predictions and softened obstacle avoidance constraints. This yields a set of candidate trajectories, and a selector evaluates their costs and picks the least suboptimal one. Finally, the FP projects the solution guess to the feasible set to ensure that the constraints are satisfied. The FP solves an NLP, minimizing the error with

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

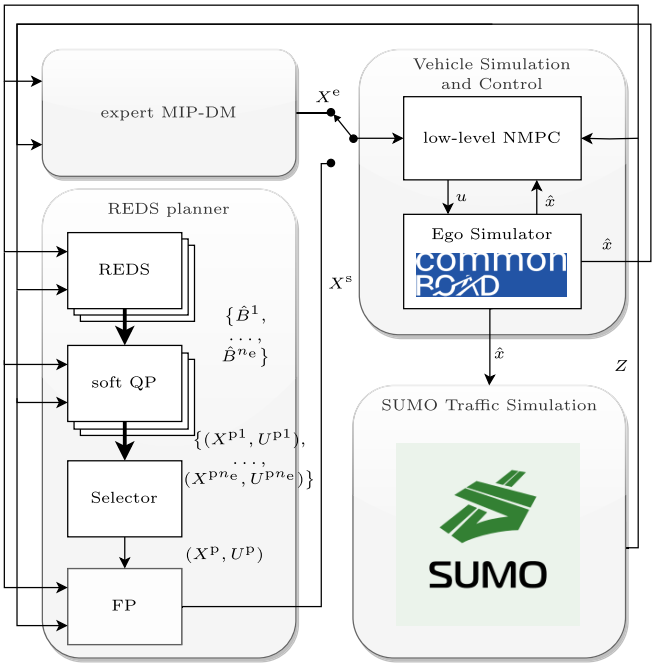

Fig. 2. Planning and closed-loop simulation architecture. The expert MIP-DM is imitated by the REDS planner, which uses an ensemble of n_e NNs to predict values of the binary variables $\{\hat{B}^1, \ldots, \hat{B}^{n_e}\}$. A soft-QP solves a formulation of the expert MIP-DM with binary variables fixed to the prediction. The lowest cost solution X^p is chosen by a selector and corrected by the FP. A reference tracking NMPC with obstacle avoidance tracks the corrected solution X^s.

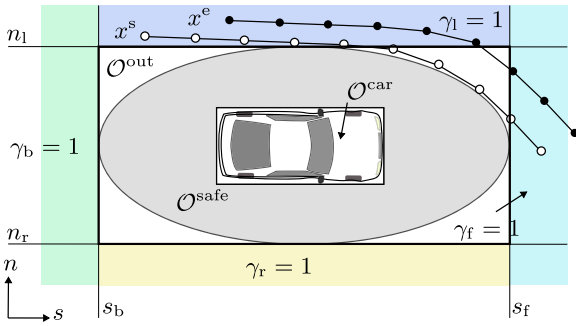

Fig. 3. Vehicle over-approximations. All trajectories outside of $\mathcal{O}^{\text{safe}}$ are considered free of collision. The expert MIP-DM plans with the most conservative obstacle set $\mathcal{O}^{\text{out}}$. The ellipsoidal smooth over-approximation $\mathcal{O}^{\text{safe}}$ is used within the FP and the NMPC. The four colored regions are uniquely determined by the binary variables γ, where in each region exactly one binary variable is equal to 1.

respect to the best candidate solution and subject to ellipsoidal collision avoidance constraints. The reference provided by the REDS planner or the expert MIP-DM is tracked by the reference tracking NMPC at a high sampling rate which includes collision avoidance constraints. This adds additional safety and leverages the requirements on the REDS planner on conservative constraints related to uncertain predictions.

The individual modules differ in motion prediction models, their objective, obstacle avoidance constraints, the approximate computation time t_{comp}, the final horizon T_f, and the discretization time t_d, see Table I.

III. Expert Motion Planner

We model the vehicle state in a curvilinear coordinate frame, which is defined by the curvature $\kappa(s)$ along the reference path [50]. The state vector $x = [s, n, v_s, v_n]^\top \in \mathbb{R}^{n_x}$ includes the position $p = [s, n]^\top \in \mathbb{R}^2$, with the longitudinal position s, the lateral position n, the longitudinal velocity v_s, and the lateral velocity v_n, where $n_x = 4$. The control vector $u = [a_s, a_n]^\top \in \mathbb{R}^{n_u}$ comprises the longitudinal and lateral acceleration in the curvilinear coordinate frame, where $n_u = 2$. The discrete-time double integrator dynamics are written as $x_{i+1} = Ax_i + Bu_i$, using the discretization time t_d, where $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$. By considering N prediction steps, the prediction horizon can be computed by $T_f = Nt_d$. Constraints $v_n \leq \overline{\alpha}\, v_s$ and $v_n \geq \underline{\alpha}\, v_s$ account for limited lateral movement of the vehicle with bounds $\underline{\alpha}, \overline{\alpha}$ that can be computed based on a maximum steering angle $\overline{\delta}$ and the maximum absolute signed curvature $\overline{\kappa} = \kappa(s^*)$, with $s^* = \arg\max_{0 \leq s \leq \overline{s}} |\kappa(s)|$ along a lookahead distance $\overline{s}$ [6]. An acceleration limit $\overline{a}_{\text{fric}}$ of the point-mass model formulated as box constraints $\|u\|_\infty \leq \overline{a}_{\text{fric}}$ inner approximate tire friction constraints related to Kamm's circle [14]. Since we formulate our model in the Frenet coordinate frame, the lateral acceleration bounds $\overline{a}_n$ and $\underline{a}_n$ are modified, based on the centrifugal acceleration, resulting in $\overline{a}_n = \overline{a}_{\text{fric}} + \overline{\kappa} v_{s,0}^2$ and $\underline{a}_n = -\overline{a}_{\text{fric}} + \overline{\kappa} v_{s,0}^2$. The fixed parameter d_{bnd} is used as the safety distance for the road boundary, bounds $\overline{u}$ and $\underline{u}$ for acceleration limits, and $\overline{v}_s$ and $\overline{v}_n$ for the maximum velocity. In order to account for the road width, bounds $\underline{n} \leq n \leq \overline{n}$ on the lateral position are used. The model used within expert MIP-DM is able to approximate the dynamics in a variety of situations [6]. However, certain maneuvers, such as sharp turns, may require additional modeling concepts [29].

A. Collision Avoidance Constraints

Collision avoidance constraints for n_{obs} obstacles are formulated by considering the ego vehicle as a point mass and using a selection matrix $P \in \mathbb{R}^{2 \times n_x}$, with $p = Px$ that selects the position states p from the states x. The true occupied obstacle space $\mathcal{O}_j^{\text{car}} \subseteq \mathbb{R}^2$, for $j \in I(n_{\text{obs}} - 1)$, is expanded to include all possible configurations, where the ego and obstacle vehicle are in a collision in the curvilinear coordinate frame, resulting in an ellipsoid $\mathcal{O}_j^{\text{safe}}$, see Fig. 3. The expert MIP-DM uses a road-frame-aligned rectangular constraint $Px_i \notin \mathcal{O}_j^{\text{out}}$, for all $i \geq 0, j \in I(n_{\text{obs}} - 1)$, which over-approximates the ellipsoidal set $\mathcal{O}_j^{\text{safe}}$, leading to additional robustness of the multilayer control architecture, with $\mathcal{O}_j^{\text{car}} \subseteq \mathcal{O}_j^{\text{safe}} \subseteq \mathcal{O}_j^{\text{out}} \subseteq \mathbb{R}^2$ (see Fig. 2 and Table I). For each obstacle j, the rectangular shape of $\mathcal{O}_j^{\text{out}}$ can be characterized by the boundaries $d^\top = [s_f, s_b, n_l, n_r]^\top$. Four collision-free regions $k \in \{f, b, l, r\}$ around an obstacle are defined, where each region can be expressed by a convex set $A^k(d)\, p \leq b^k(d)$, with either one for $k \in \{f, b\}$ or three half-space constraints for $k \in \{l, r\}$, see Fig. 3. Four binary variables $\gamma = [\gamma_f, \gamma_b, \gamma_l, \gamma_r]^\top \in \mathbb{B}^4$ are used with a big-M formulation to ensure that the vehicle is inside one of the four regions. Therefore,with $\mathbf{1}$ in the according dimension of

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

TABLE I
OVERVIEW OF OPTIMIZATION PROBLEMS SOLVED WITHIN THE INDIVIDUAL MODULES

Module	Opt. Class	Model	Objective	Obstacle Avoidance	t_{comp}	T_{f}	t_{d}	Comment
expert MIP-DM	MIQP	point-mass	global economic	hyper-planes $\mathcal{O}^{\text{out}}$	$\sim$ 2s	10s	0.2s	high computation time
soft-QP	QP	point-mass	global economic	fixed hyper-planes $\mathcal{O}^{\text{out}}$	$\sim$ 3ms	10s	0.2s	possibly unsafe
FP	NLP	point-mass	tracking (soft-QP)	ellipsoids $\mathcal{O}^{\text{safe}}$	$\sim$ 10ms	10s	0.2s	safe projection
tracking NMPC	NLP	kinematic	tracking (FP)	ellipsoids $\mathcal{O}^{\text{safe}}$	$\sim$ 2ms	1s	0.05s	

either $\mathbf{1}_1$ for $k \in \{f, b\}$ or $\mathbf{1}_3$ for $k \in \{l, r\}$, the collision-free set is

$$\mathcal{F}^{\text{out}}(d, \overline{\sigma}) = \big\{ (p, \gamma, \sigma) \in \big(\mathbb{R}^2, \mathbb{B}^4, \mathbb{R}\big) \big| \forall k \in \{f, b, l, r\} : A^k(d)\, p \le b^k(d) + \mathbf{1}(1 - \gamma_k)M + \mathbf{1}(\sigma - 1)\overline{\sigma}_k,\ \mathbf{1}_4^\top \gamma = 1 \big\}. \quad (1)$$

The values $\overline{\sigma}_k$ define additional safety margins for each of the four collision-free regions. The slack variable $\sigma \in \mathbb{R}$ is bounded $0 \le \sigma \le 1$, such that only points in the exterior of $\mathcal{O}_j^{\text{out}}$ satisfy the condition in (1). A model predicts the future positions of the obstacle boundaries d_i^j for each obstacle $j \in I(n_{\text{obs}} - 1)$ at time steps $i \in I(N)$ along the horizon.

B. MIQP Cost Function

The MIQP cost comprises quadratic penalties for the state vector $x \in \mathbb{R}^{n_x}$ with weight $Q \in \mathbb{R}^{n_x \times n_x}$, and the control vector $u \in \mathbb{R}^{n_u}$ with weight $R \in \mathbb{R}^{n_u \times n_u}$, for tracking of their references $\tilde{x}$ and $\tilde{u}$, respectively. The reference $\tilde{x}_i = [\tilde{s}_i, \tilde{n}_i, \tilde{v}_s, 0]^\top$ at time step i is determined by the desired velocity $\tilde{v}_s$, as well as by the binary control vector $\lambda = [\lambda^{\text{up}}, \lambda^{\text{down}}]^\top \in \mathbb{B}^2$. These binary variables are used to determine lane changes at time step i, resulting in the road-aligned lateral reference $\tilde{X}_n = [\tilde{n}_0, \ldots, \tilde{n}_N]^\top \in \mathbb{R}^{N+1}$ always being the center of the target lane. The longitudinal position $\tilde{s}_i$ follows from the velocity $\tilde{v}_s$. The MIQP cost function reads

$$\sum_{i=0}^{N} \|x_i - \tilde{x}_i\|_Q^2 + \sum_{i=0}^{N-1} \|u_i - \tilde{u}_i\|_R^2 + w_{\text{lc}} \sum_{i=0}^{N-1} \mathbf{1}_2^\top \lambda_i + w_{\text{rght}} \sum_{i=0}^{N} n_i + w_{\text{dst}} \sum_{j=0}^{n_{\text{obs}}-1} \sum_{i=0}^{N} \left(\sigma_i^j\right)^2 \quad (2)$$

including a penalty with weight $w_{\text{rght}} > 0$ to minimize deviations from the right-most lane, a weight $w_{\text{lc}} > 0$ to penalize lane changes, and a weight $w_{\text{dst}} > 0$ penalizing slack variables to avoid being too close to any obstacle.

C. Parametric MIQP Formulation

For a horizon of N steps, the binary MIQP decision variables are $\Lambda = [\lambda_0, \ldots, \lambda_{N-1}] \in \mathbb{B}^{2\times N}$ and $\Gamma = [\gamma_0^0, \gamma_1^0, \ldots, \gamma_N^{n_{\text{obs}}-1}] \in \mathbb{B}^{4\times n_{\text{obs}}(N+1)}$, the real-valued states are $X = [x_0, \ldots, x_N] \in \mathbb{R}^{4\times(N+1)}$, the control inputs are $U = [u_0, \ldots, u_{N-1}] \in \mathbb{R}^{2\times N}$, and the slack variables are $\Sigma = [\sigma_0^0, \sigma_1^0, \ldots, \sigma_N^{n_{\text{obs}}-1}] \in \mathbb{R}^{n_{\text{obs}}(N+1)}$. Hard linear inequality constraints are

$$\begin{aligned} &\{H(X, U) \ge 0\} \\ &\quad \Leftrightarrow \big\{X, U \mid \underline{u} \le u_i \le \overline{u}, i \in I(N-1) \\ &\qquad \underline{x} \le x_i \le \overline{x}, \underline{\alpha}\, v_{s,i} \le v_{\text{n},i} \le \overline{\alpha}\, v_{s,i}, i \in I(N)\big\}. \end{aligned}$$

The parameters $\Pi = (\hat{x}, \tilde{v}_s, n_{\text{lanes}}, D)$ include the initial ego state $\hat{x}$, the desired velocity $\tilde{v}_s$, the number of lanes n_{lanes}, and the time-dependent obstacle bounds D, where

$$D = \left\{ d_i^j \;\middle|\; \forall j \in I(n_{\text{obs}} - 1)\ \forall i \in I(N) \right\}.$$

The parametric MIQP solved within each iteration of the MIP-DM is

$$\min_{X, U, \tilde{X}_n, \Lambda, \Gamma, \Sigma} J^e\big(X, U, \tilde{X}_n, \Lambda, \Sigma\big) \quad (3a)$$

$$\text{s.t. } x_0 = \hat{x}, \quad H(X, U) \ge 0,\ 0 \le \Sigma \le 1 \quad (3b)$$

$$\tilde{n}_{i+1} = \tilde{n}_i + d_{\text{lane}} \lambda_i^{\text{up}} - d_{\text{lane}} \lambda_i^{\text{down}} \quad (3c)$$

$$x_{i+1} = Ax_i + Bu_i, \quad i \in I(N-1) \quad (3d)$$

$$\left(P x_i, \gamma_i^j, \sigma_i^j\right) \in \mathcal{F}^{\text{out}}\left(d_i^j, \overline{\sigma}_i^j\right), \quad i \in I(N)$$
$$j \in I(n_{\text{obs}} - 1) \quad (3e)$$

where the cost $J^e(\cdot)$ is defined in (2) and $\tilde{n}_0$ is the lateral position of the center of the desired lane. An MIQP solving (3) is used as an "expert" to collect supervisory data, i.e., feature-label pairs (Π, B^*), where B^* is the optimal value of binary variables, see Fig. 2. For the closed-loop evaluation of the expert MIP-DM, the optimizer X^* is used as the output of the expert MIP-DM X^e.

IV. SCALABLE EQUIVARIANT DEEP NEURAL NETWORK

Because the MIQP (3) is computationally demanding to solve in real-time, especially for long prediction horizons and a large number of obstacles, we propose a novel variant of the combinatorial offline convex online (COCO) algorithm [7] to accelerate MIQP solutions using supervised learning. We train an NN to predict binary variables and then solve the remaining convex QP online, after fixing the binary variables. The MIQP (3) comprises $4n_{\text{obs}}N$ structured binary variables related to obstacles and $2N$ lane change variables. We refer to the latter as *unstructured* binary variables because, differently from the others, there is no specific relation besides recurrence among them.

In the following, we first describe the desired properties of the NN prediction in Section IV-A and review the classification of binary variables in Section IV-B. Then,

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

in Section IV-C, we introduce one of our main contributions, the REDS to achieve the desired properties. Finally, we show in Section IV-D how to use an ensemble of NNs to generate multiple predictions.

A. Desired Predictor Properties for Motion Planning

The desired properties of the predictor may be divided into performance, i.e., general metrics that define the NN prediction performance, and structural properties, i.e., structure-exploiting properties related to Requirements 1–5.

1) Performance: The prediction performance is quantified by the likelihood of predicting a feasible solution μ, and a measure of optimality ρ. However, supervised learning optimizes accuracy, i.e., cross-entropy loss, which was shown to correlate well with feasibility [7], [8], [12]. In fact, we evaluated the Pearson correlation coefficient (PCC) for our experiments and obtained a PCC of 0.81 for the correlation between the training loss and the infeasibility and a PCC of 0.88 between accuracy and feasibility. In addition, the computation time t_{comp} to evaluate the NN is important for real-time feasibility. We aim for t_{comp} to be very small compared with the MIQP solution time, and $t_{\text{comp}} < t_p \leq 0.2$ s (see Requirement 4). Finally, the memory footprint of the NN should be small for implementation on embedded microprocessors [4].

2) Structure: First, the REDS planner needs to operate on a variable number of obstacles, which is required in real traffic scenarios, see Requirement 5. Second, to comply with Requirement 3, obstacle-related predictions need to be equivariant to permutations on the input, see Definition 2. For unstructured binary variables, the predictions should be permutation invariant, see Definition 1. Third, again relating to Requirement 5, the NN architecture is expected to generalize to unseen data. Particularly, it should provide accurate predictions for several obstacles that may not be present in the training data. Finally, the NN should predict multiple guesses to increase the likelihood of feasibility and/or optimality. The proposed REDS planner provides the desired structural properties, and it improves the performance properties.

Since we consider a highly structured problem domain, we propose to directly include the known structure into the NN architecture. Alternatively, one could learn the structure for general problems by, e.g., attention mechanisms [51]. However, the related attention-based architectures usually have a high inference time which may clash with the desired fast online evaluation.

B. Prediction of Binary Variables by Classification

As [12] and [37] show, solving the prediction of binary variables as a multiclass classification problem, yields superior results than solving it as a regression problem. Since the naive enumeration of binary assignments grows exponentially, i.e., the number of assignments is $2^{|B|}$, where $|B|$ is the number of binary variables, an effective strategy is to enumerate only combinations of binary assignments that actually appear in the dataset. While it cannot guarantee to predict all combinations, this leads to a much smaller number of possible classes and it has been observed that the resulting classifications still significantly outperform regression (see [12]).

C. Recurrent Equivariant Deep Set Architecture

The REDS architecture, as shown in Fig. 4, achieves the structural properties and improves the prediction performance. We use training data that consists of feature-label pairs (Π, B). The features Π are split into obstacle-related features $\zeta^{\text{eq}} = \{\zeta_0^{\text{eq}}, \ldots, \zeta_{n_{\text{obs}}-1}^{\text{eq}}\}$, where $\zeta_i^{\text{eq}} \in \mathbb{R}^{m_{\text{eq}}}$, and unstructured features $\zeta^{\text{us}} \in \mathbb{R}^{m_{\text{us}}}$. The equivariant features $\zeta^{\text{eq}} = (z_j, d_j)$ are the *initial* obstacle state z_j and its spatial dimension d_j, since all states along the horizon are predicted based on the initial state. The unstructured features contain all other parameters of Π, i.e., $\zeta^{\text{us}} = (\hat{x}, \tilde{v}_s, n_{\text{lanes}})$.

Zaheer et al. [49] propose a simple but effective NN architecture that provides either permutation equivariance or invariance. The blocks are combined in our tailored *encoder layer* that maintains permutation equivariance for the equivariant outputs and invariance for the unstructured outputs, see Fig. 4. A hidden state $h^{\text{us}} \in \mathbb{R}^{m_h}$ is propagated for unstructured features and hidden states $h_j^{\text{eq}} \in \mathbb{R}^{m_h}$, with $j \in I(n_{\text{obs}} - 1)$, for the equivariant features, where $h^{\text{eq}} = [h_0^{\text{eq}}, \ldots, h_{n_{\text{obs}}-1}^{\text{eq}}]^\top \in \mathbb{R}^{n_{\text{obs}} \times m_h}$. The encoder layer has four directions of information passing between the fixed-size unstructured and the variable-size equivariant hidden states with input dimension m_h and output dimension m_h':

1) *Equivariant to Equivariant:* Equivariant deep sets [49] are used as layers with rectified linear unit (ReLU) activation functions $\sigma(\cdot)$ and parameters $\Theta^{\text{ee}}, \Gamma^{\text{ee}} \in \mathbb{R}^{m_h \times m_h'}$

$$f^{\text{ee}}\left(h^{\text{eq}}\right) = \sigma\left(h^{\text{eq}}\Theta^{\text{ee}} + \mathbf{1}\mathbf{1}^\top h^{\text{eq}}\Gamma^{\text{ee}}\right).$$

2) *Equivariant to Unstructured:* To achieve invariance from the set elements h_j^{eq} to the unstructured hidden state h^{us}, the invariant layer of [49] is added that sums up over the set elements with parameters $\Theta^{\text{eu}} \in \mathbb{R}^{m_h \times m_h'}$

$$f^{\text{eu}}\left(h^{\text{eq}}\right) = \sigma\left(\left(\sum_{j=0}^{n_{\text{obs}}-1} h_j^{\text{eq}}\right)\Theta^{\text{eu}}\right).$$

3) *Unstructured to Equivariant:* To implement a dependency of the equivariant elements on the unstructured hidden state while maintaining equivariance, this layer equally influences each set element by

$$f^{\text{ue}}\left(h^{\text{us}}\right) = \sigma\left(\mathbf{1}_{n_{\text{obs}}} \otimes h^{\text{us}}\Theta^{\text{ue}}\right)$$

with parameters $\Theta^{\text{ue}} \in \mathbb{R}^{m_h \times m_h'}$.

4) *Unstructured to Unstructured:* We use a standard feed-forward (FF) layer with parameters $\Theta^{\text{uu}} \in \mathbb{R}^{m_h \times m_h'}$

$$f^{\text{uu}}\left(h^{\text{us}}\right) = \sigma\left(h^{\text{us}}\Theta^{\text{uu}}\right).$$

FF networks act as encoders to the input features, which allows to matching the dimensions of the equivariant and unstructured hidden states. For each layer of the REDS in Fig. 4, the contributions are summed up to obtain the new hidden states

$$h^{\text{eq}\prime} = f^{\text{ee}}\left(h^{\text{eq}}\right) + f^{\text{ue}}\left(h^{\text{us}}\right)$$
$$h^{\text{us}\prime} = f^{\text{uu}}\left(h^{\text{us}}\right) + f^{\text{eu}}\left(h^{\text{eq}}\right).$$

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

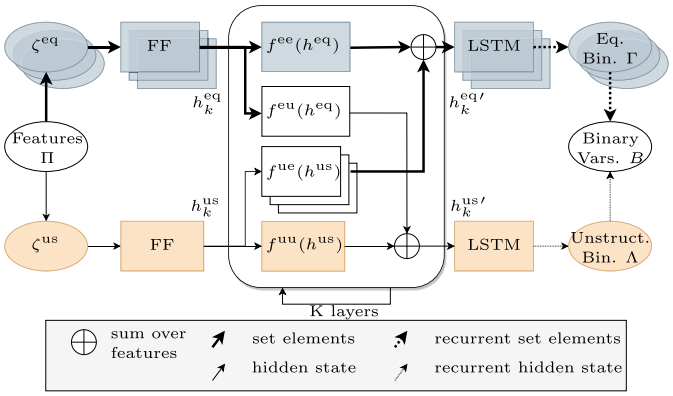

Fig. 4. REDS network. The blue blocks show the propagation of equivariant features, whereas the orange blocks show the propagation of unstructured features. An invariant connection couples both hidden states.

A long short-term memory (LSTM) is used as *decoder* for each equivariant hidden state, transforming the hidden state into a time series of binary predictions, see Fig. 4. Another LSTM is used as a *decoder* for unstructured hidden states. For the REDS, the classification problem per time step (Section IV-B) needs to consider only four classes per obstacle (one for each collision-free region), see Fig. 3, and three classes for lane changes (change to the left or right, stay in lane).

D. Neural Network Ensembles for Multiple Predictions

As suggested in early works [52], [53], using an ensemble of n_e stochastically trained NNs is a simple approach to obtain multiple guesses and improve classification accuracy. Producing multiple predictions, the lowest cost maneuver among different candidate solutions can be selected, e.g., staying behind a vehicle or overtaking. For typical classification tasks, no a posteriori oracle exists that identifies the *best* guess [52]. However, in our problem setup, the soft-QP solution can directly evaluate the feasibility and optimality and therefore, identify the best guess, as discussed next.

V. Soft QP Solution and Selection Method

For each candidate solution from the ensemble of NNs, the soft-QP is constructed based on the MIQP (3) with fixed binary variables and unbounded slack variables for the obstacle constraints. Each candidate solution consists of the binary values $\hat{\Lambda} = [\hat{\lambda}_0, \dots, \hat{\lambda}_{N-1}]$ and $\hat{\Gamma} = [\hat{\gamma}_0^0, \hat{\gamma}_1^0, \dots, \hat{\gamma}_N^{n_{\text{obs}}-1}]$ that are predicted by the NN. The reference $\bar{X}_n$ in (3c) is also fixed when the binary variables are fixed. The resulting soft-QP is convex, and a feasible solution always exists due to removing the upper bound for each slack variable σ_i^j, with $j \in I(n_{\text{obs}}-1)$, $i \in I(N)$ and

$$J^{s*} = \min_{X,U,\Sigma} J^s(X,U,\Sigma) \tag{4a}$$

$$\text{s.t. } x_0 = \hat{x}, \quad H(X,U) \geq 0, \ \Sigma \geq 0 \tag{4b}$$

$$x_{i+1} = Ax_i + Bu_i, \quad i \in I(N-1) \tag{4c}$$

$$\left(Px_i, \sigma_i^j\right) \in \mathcal{F}^{\text{out}}\left(d_i^j, \hat{\gamma}_i^j\right), \quad i \in I(N)$$

$$j \in I(n_{\text{obs}}-1) \tag{4d}$$

where the cost $J^s(\cdot)$ is (2), see Table I. We construct (4d) from (1) by removing the safety margins $\overline{\sigma}_i^j$ and fixing the binary variables to the predicted solution guess $\hat{\Gamma}$. Therefore, the soft-QP (4) is a relaxation of MIQP (3) with fixed integers. Problem (4) is solved for each prediction of the NN ensemble, and the solution leading to the lowest cost for (4a) is selected as output X^p of the module, see Fig. 2. The soft-QP is convex and can be solved efficiently using a structure exploiting QP solver [9], [54]. Despite solving multiple QPs for multiple candidate solutions, the computational burden is much lower than solving an MIQP that typically requires solving a combinatorial amount of convex relaxations.

VI. Feasibility Projection and SQP Algorithm

The soft-QP solution (X^p, U^p) may not be collision-free due to binary classification errors from the ensemble of NNs, i.e., some of the slack variables may be nonzero in the soft-QP solution. In order to project the soft-QP solution to a collision-free trajectory, a smooth convex–concave NLP, referred to as FP, is solved in each iteration. The FP solves an optimization problem that is similar to (4) (see Table I), but the reference trajectory is equal to the soft-QP solution, i.e., $\tilde{X} := X^p$ and $\tilde{U} := U^p$. In addition, the obstacle constraints in (4d) are replaced by smooth concave constraints based on the ellipsoidal collision region, see Fig. 3, which allows the use of an efficient SQP algorithm, see [55].

A. Optimization Problem Formulation

With the geometry parameter d of an obstacle, the inner ellipse $\mathcal{O}^{\text{safe}}$ within $\mathcal{O}^{\text{out}}$ is used for defining the safe-set $\mathcal{F}^{\text{safe}}$. The ellipse center and axis matrix

$$t(d) = \frac{1}{2}\left[\left(s_f + s_b\right), \left(n_l + n_r\right)\right]^\top$$

$$\Upsilon(d) = \frac{1}{\sqrt{2}}\text{diag}\left(\left[s_f - s_b, n_l - n_r\right]\right)$$

are used to formulate the smooth obstacle constraint

$$\mathcal{F}^{\text{safe}}(d) = \left\{(p,\xi) \middle| \|p - t(d)\|^2_{\Upsilon^{-1}(d)} \geq 1 - \xi \right\} \tag{5}$$

with slack variables $\xi \geq 0$. A tracking cost

$$J_{\text{tr}}^f(X,U) = \sum_{i=0}^{N} \left\|x_i - \tilde{x}_i\right\|_Q^2 + \sum_{i=0}^{N-1} \left\|u_i - \tilde{u}_i\right\|_R^2 \tag{6}$$

and a slack violation cost

$$J_{\text{slack}}^f(\Xi) = w_h \sum_{j=0}^{n_{\text{obs}}-1} \sum_{i=0}^{N} \xi_i^j \tag{7}$$

are defined, where $\Xi = \{\xi_i^j | i \in I(N), j \in I(n_{\text{obs}}-1)\}$. The penalty $w_h \gg 0$ is sufficiently large such that a feasible solution with $\xi_i^j = 0$ can be found when it exists. The resulting NLP can be written as follows:

$$\min_{X,U,\Xi} J_{\text{tr}}^f(X,U) + J_{\text{slack}}^f(\Xi) \tag{8a}$$

$$\text{s.t. } x_0 = \hat{x}, \quad H(X,U) \geq 0, \ \Xi \geq 0 \tag{8b}$$

$$x_{i+1} = Ax_i + Bu_i, \quad i \in I(N-1) \tag{8c}$$

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

$$\left(Px_i, \xi_i^j\right) \in \mathcal{F}^{\text{safe}}\left(d_i^j\right), \quad i \in I(N)$$
$$j \in I(n_{\text{obs}} - 1) \tag{8d}$$

using the least-squares tracking cost (6) and smooth obstacle avoidance constraints (5). The optimal trajectory X^* of (8) is the output of the FP X^s and also of the REDS planner and tracked by the NMPC, see Fig. 2. Since $\mathcal{F}^{\text{out}} \subseteq \mathcal{F}^{\text{safe}}$, the NLP (8) is a smooth relaxation of the MIQP (3).

B. Application of Sequential Quadratic Programming

NLP (8) has a convex–concave structure. Except for the concave constraints (8d), the problem can be formulated as a convex QP. When solving NLP (8) using the Gauss–Newton SQP method [56], this guarantees a bound on the constraint violation for the solution guess at each SQP iteration [10].

Proposition 1: Consider NLP (8), let X_i, U_i, and Ξ_i be the primal variables after an SQP iteration with Gauss–Newton Hessian approximation, and let X_0, U_0, and Ξ_0 be an initial guess equal to the reference, i.e., $X_0 = \tilde{X}$, $U_0 = \tilde{U}$. Then, the decrease in the slack cost reads

$$J^f_{\text{slack}}(\Xi_i) \leq J^f_{\text{slack}}(\Xi_0) - J^f_{\text{tr}}(X_i, U_i). \tag{9}$$

Proof: According to [10, Lemma 4.2] cost of (8) decreases after each iteration for our problem structure, i.e.,

$$J^f_{\text{tr}}(X_{i+1}, U_{i+1}) + J^f_{\text{slack}}(\Xi_{i+1}) \leq J^f_{\text{tr}}(X_i, U_i) + J^f_{\text{slack}}(\Xi_i).$$

Consequently, (9) can be verified, since it holds that $J^f_{\text{tr}}(X_0, U_0) = 0$ due to the initialization equal to the reference and $J^f_{\text{tr}}(X, U) \geq 0, \forall X, U$, such that

$$J^f_{\text{tr}}(X_i, U_i) + J^f_{\text{slack}}(\Xi_i) \leq J^f_{\text{slack}}(\Xi_0).$$

□

In addition, it can be guaranteed that the SQP iterations remain feasible, i.e., collision-free, once a feasible solution is found, if the slack weights $w_h \gg 0$ are chosen *sufficiently large*. The latter requires to select a weight value such that the gradient of the *exact* $L1$ penalty (7) is larger than any gradient of the quadratic cost (6) in the bounded feasible domain. This property is due to the exact penalty formulation [57] and the inner approximations of the concave constraints (8d), since a feasible linearization point in iteration j guarantees feasibility also in the next iterate $j + 1$ [55]. Notably, choosing sufficiently large weights, yet not too large to avoid ill-conditioned QPs, may be challenging in practice. Therefore, the weights could be increased in each iteration, if convergence issues were encountered [57]. However, we used constant weights without encountering numerical problems.

Under mild assumptions [10], the SQP method converges to a stationary point of (8). The FP provides a certificate of feasibility if the slack variables are 0, i.e., $\Xi_j = 0$. We show that the FP effectively increases the likelihood of computing a collision-free trajectory in numerical simulations.

VII. Implementation Details

In this section, we list the most important implementation details. Further information on the structure of the NN model, training parameters, and the FP are in the Appendix.

A. Numerical Solvers

For solving the MIQP of the expert MIP-DM and the QP of the soft-QP, we use `gurobi` [11] and formulate both problems in `cvxpy` [58]. The NLPs of the FP, as well as the reference tracking NMPC, are solved by using the open-source solver `acados` [9] and the algorithmic differentiation framework `casadi` [59]. We use Gauss–Newton Hessian approximations, full steps, an explicit RK4 integrator, and noncondensed QPs that are solved by `HPIPM` [54]. We use real-time iterations [25] within the reference tracking NMPC and limit the maximum number of SQP iterations to 10 within the FP.

B. Training of the NN

In order to formulate and train the NNs, we use `PyTorch`. We train on datasets of 10^5 expert trajectories that we generate by solving the expert MIP-DM with randomized initial parameters, sampled from a uniform distribution within the problem bounds, see the Appendix. We use a learning rate of 10^{-4} and a batch size of 1024. The performance is evaluated on a test dataset of 2×10^3 samples using a *cross-entropy loss* and the `adam` optimizer [60].

C. Computations

Simulations are executed on a `LENOVO ThinkPad L15 Gen 1` Laptop with an `Intel`[1] `Core`[2] `i7-10510U at 1.80 GHz` CPU. The training and GPU evaluations of the NNs are performed on an Ubuntu workstation with two `GeForce RTX 2080 Ti PCI-E 3.0 11264-MB GPUs`. Parts of the REDS planner, namely the n_e NN ensemble and the soft-QP, can be parallelized, which speeds up our approach by approximately the number of NNs used. Therefore, the *serial* computation time

$$t_s = \sum_{i=1}^{n_e} \left(t_{\text{NN},i} + t_{\text{QP},i}\right) + t_{\text{FP}}$$

includes each individual NN inference time $t_{\text{NN},i}$, each soft-QP evaluation time $t_{\text{QP},i}$ and the FP evaluation time t_{FP}. The *parallel* computation time is

$$t_p = \max_{i=1,\ldots,n_e} \left(t_{\text{NN},i} + t_{\text{QP},i}\right) + t_{\text{FP}}.$$

D. Nonlinear Model Predictive Control

The lower level reference tracking NMPC is formulated as shown in [50], which is similar to (8), but using a more detailed nonlinear kinematic vehicle model, a shorter sampling period and a shorter horizon (see Table I). The reference tracking NMPC can be solved efficiently using the real-time iterations [25], based on a Gauss–Newton SQP method [56] in combination with a structure exploiting QP solver [9].

[1]Registered trademark.
[2]Trademarked.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

VIII. In-Distribution Evaluations

First, we show the performance of the REDS architecture, compared with the state-of-the-art architectures of [12], and we demonstrate its *structural properties*, see Section IV-A2. Next, we show how the soft-QP and the feasibility projection increase the prediction performance and the influence on the overall computation time. In this section, we use the same distribution over the input parameters for training and testing of the NNs.

A. Evaluation of the REDS

We compare two variants of the proposed architecture. First, the REDS is evaluated, see Fig. 4. Second, as an ablation study, the same architecture but without the LSTM, referred to as equivariant deep set (EDS), is evaluated. In the EDS, an FF is used to predict the binary variables along the full prediction horizon (see Fig. 4). The performance is compared against the state-of-the-art architectures for similar tasks [12], i.e., an LSTM and an FF network with a comparable amount of parameters.

The evaluation metrics are the infeasibility rate, i.e., the share of infeasible soft-QPs violating constraints (with nonzero slack variables), and the misclassification rate, i.e., the share of wrong classifications concerning the prediction of any binary variable. If at least one binary variable in the prediction is wrongly classified, the whole prediction is counted as misclassified, even though the soft-QP computes a feasible low-cost solution. Furthermore, we consider the suboptimality ρ, i.e., the objective of the feasible soft-QP solutions J^{s*} compared with the expert MIP-DM cost J^{e*}

$$\rho = \frac{J^{s*} - J^{e*}}{J^{e*}} \geq 0. \tag{10}$$

A suboptimality of $\rho = 0$ means the cost of the prediction is equal to the optimal cost of the expert MIP-DM. Fig. 5 shows how the performance scales with the number of obstacles n_{obs} and with the horizon length N, without the FP from Section VI. The REDS and EDS yield superior results to the LSTM as soon as $n_{\text{obs}} \geq 2$, and they vastly outperform the FF network for an increasing number of obstacles and horizon length.

Besides improving the *performance* metrics, the REDS also provides the desired structural properties. An REDS network can be trained and evaluated with a variable number of obstacles and prediction horizon. In Table II, the generalization performance of REDS network is shown, i.e., it is evaluated for number of obstacles that were not present in the training data. Table II lists the comparison of generalization for an *interpolation* and *extrapolation* of the number of obstacles and the prediction horizon. According to Table II, REDS generalizes well for samples out of the training data distribution for the number of obstacles and the horizon length.

B. Evaluation for Ensemble of REDS Networks

In Fig. 6, we show the achieved feasibility rate on the test data using an ensemble with $n_e = 10$ REDS networks. In total, 2×10^3 samples are evaluated for each NN individually and cumulatively. The performance improves by adding more NNs, however, also the total computation time increases. The parallel computation time results in the maximum time over the individual networks. Table III shows the memory usage and the number of parameters for different architectures. The sizes of the networks are feasible for embedded devices [4], i.e., the REDS and EDS provide improved the performance with a comparable or even reduced memory footprint than FF networks and LSTMs. This may be due to exploiting the structure of equivariances and invariances inherently occurring in the application. Similar observations were made in other applications, where deep sets have been applied, see [42].

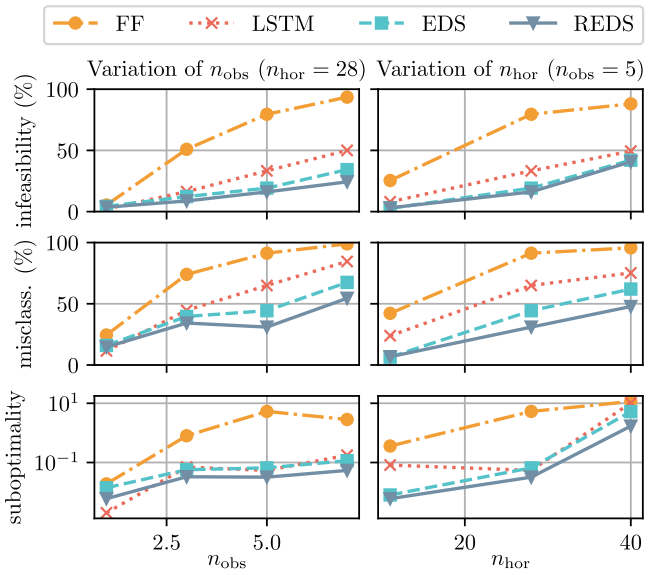

Fig. 5. Performance evaluation for infeasibility rate, misclassification rate, and suboptimality of different network architectures, depending on the number of obstacles and horizon length. The REDS network outperforms the other architectures, particularly for a larger number of obstacles and a longer horizon. Suboptimality is shown in a logarithmic scale.

TABLE II
Evaluation of the REDS Network Generalization Performance (Highlighted in Bold)

	Training		Testing		Performance (%)		
	n_{obs}	N	n_{obs}	N	misclass.	infeas.	subopt.
Generalization for the number of obstacles: *interpolation*							
No general.	1-5	28	3	28	41.7	19.7	17.4
General.	1,2,4,5	28	3	28	**42.0**	**19.8**	**11.6**
Generalization for the number of obstacles: *extrapolation*							
No general.	1-5	28	5	28	30.3	24.4	3.9
General.	1-4	28	5	28	**34.4**	**26.5**	**3.4**
Generalization for the horizon length							
No general.	1-3	16	1-3	16	20.4	8.9	38.1
General.	1-3	12	1-3	16	**26.0**	**10.8**	**20.9**

C. Evaluation of REDS Planner With Feasibility Projection

The REDS planner is validated on random samples of the test data, i.e., samples of the same distribution as the NN training data. Fig. 7 shows the infeasibility rate and suboptimality (10) of the REDS planner, using either the QP

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

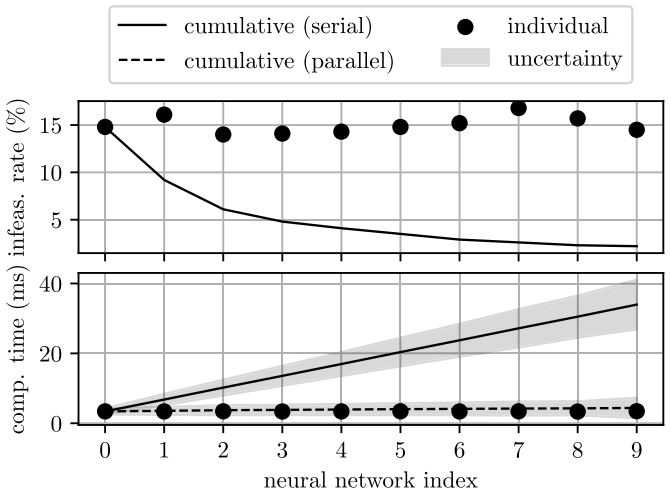

Fig. 6. Individual and cumulative REDS prediction performance (infeasibility rate) and computation time concerning the number of NNs used in the ensemble. The computation time differs for parallel and serial evaluation. For the parallel evaluation, the slowest network determines the computation time.

TABLE III
MEMORY USAGE AND NUMBER OF PARAMETERS IN NN ARCHITECTURES INVOLVING FIVE OBSTACLES AND A HORIZON OF 28 STEPS

	FF	LSTM	EDS	REDS
Memory usage (MB)	2.40	0.97	0.51	1.40
Number of parameters (10^5)	5.98	2.41	1.26	3.43

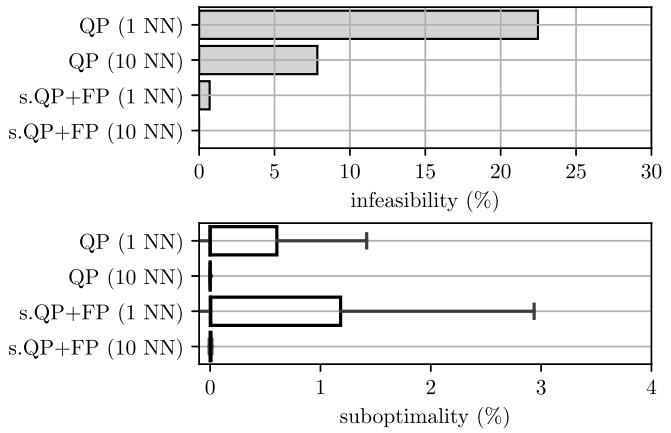

Fig. 7. Open-loop comparison of infeasibility rate and suboptimality of the REDS planner, using the QP without slack variables and the soft-QP followed by the FP. Infeasible problems are not considered in the suboptimality.

without slack variables or the soft-QP followed by the FP. For an ensemble of ten NNs, the infeasibility rate of the QP without slack variables is below 10% and decreased to almost 0% by using the soft-QP followed by the FP, and the suboptimality is also negligible. The suboptimality of the soft-QP followed by the FP is higher than the suboptimality of the QP, since also infeasible problems are rendered feasible, yet with increased suboptimality values. Fig. 8 shows the box plots of the computation times related to different components. The main contributions to the total computation time of the REDS planner stem from the soft-QP (median of ∼4.2 ms)

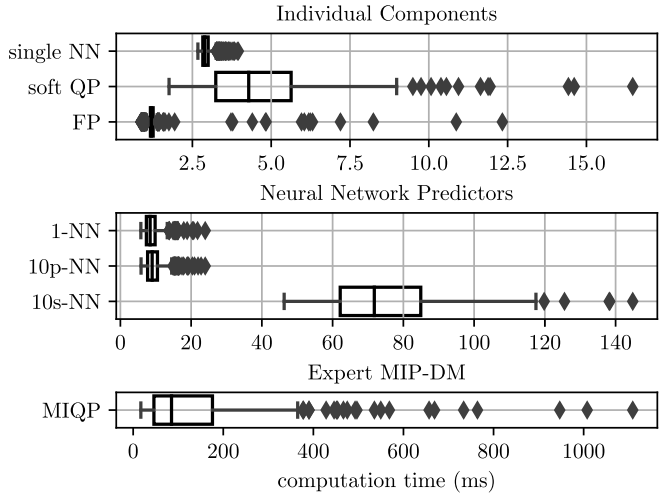

Fig. 8. Box plots for open-loop comparison of the computation times of 10^3 samples. REDS planner with an ensemble of one (1-NN) or ten NNs is parallelized (10p) or serial (10s).

and the NNs (median of ∼3.0 ms per network). While the parallel architecture with ten NNs, as well as a single NN, decrease the worst case MIQP computation time by a factor of approximately 50 and the median by a factor of 10, the serial approach decreases the maximum by a factor of 8 and the median by a factor of 2. Besides computation time, the REDS planner is suitable for embedded system implementation as opposed to high-performance commercial solvers like the one used here as an expert, i.e., `gurobi` [11].

IX. CLOSED-LOOP VALIDATIONS WITH SUMO SIMULATOR

The following closed-loop evaluations of the REDS planner on a multilane highway scenario yield a more realistic performance measure. This involves challenges such as the distribution shift of the input parameters, wrong predictions, and errors of the reference tracking NMPC.

A. Setup for Closed-Loop Simulations

Several choices need to be made for parameters in the REDS planner and in the simulation environment.

1) Collision Avoidance: For an environment with a large number of obstacles, we implement the following heuristic to select up to $n_{\text{obs}} = 5$ obstacles in proximity to the ego vehicle. We consider the closest successive obstacles in each lane that is not the lane of the ego vehicle and the leading vehicle on the ego lane. In Fig. 9, the selection of obstacles in proximity to the ego vehicle is shown in the Frenet coordinate frame. Obstacles are plotted in consecutive planner time steps, and the color indicates whether they are considered at each time step. Overtaking is allowed on both sides of a leading vehicle.

2) Sampling Frequency: The planning frequency is set to 5 Hz, the control and ego vehicle simulation rate is 50 Hz, and the `SUMO` simulation is 10 Hz. The traffic simulator in `SUMO` is slower than the ego vehicle simulation frequency; therefore, the motion of the vehicles is linearly extrapolated between

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

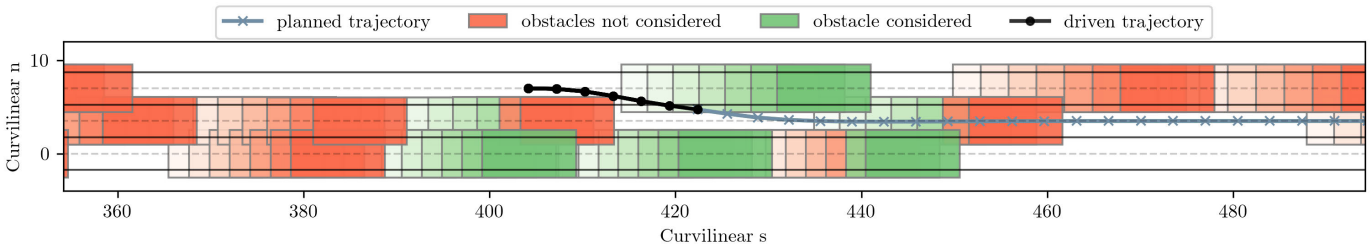

Fig. 9. Obstacles considered at each planning step in the Frenet coordinate frame. The plot shows the ego point-mass trajectory and the inflated obstacles in proximity to the ego vehicle for seven-time steps. The obstacle color indicates whether it was considered in planning or not.

SUMO updates. Indeed, the REDS planner is real-time capable for selected frequencies, see Requirement 4.

3) Vehicle Models: We use parameters of a *BMW 320i* for the ego vehicle, which is a medium-sized passenger vehicle whose parameters are provided in CommonRoad [14] for models of different fidelities. An *odeint* integrator of scipy simulates the 29-state *multibody model* [14] with a 20-ms time step. The traffic simulator SUMO [13] simulates interactive driving behaviors with the *Krauss model* [61] for car following and the *LC2013* [62] model for lane changing. From zero up to five surrounding vehicles are selected for our comparisons.

4) Road Layout: Standardized scenarios on German roads, provided by the *scenario database* of CommonRoad, are fully randomized before each closed-loop simulation, including the start configuration of the ego vehicle and all other vehicles. This initial randomization, in addition to the interactive and stochastic behavior simulated in SUMO, covers a wide range of traffic situations, including traffic jams, blocked lanes, and irrational driver decisions, such as half-completed lane changes. The basis of our evaluations is the three-lane scenario DEU_Cologne-63_5_I-1 with all (*dense*) or only a fourth (*sparse*) of the vehicles in the database.

B. Distribution Shift

The generally unknown state distribution encountered during closed-loop simulations, referred to as simulation distribution (SD), differs from the training distribution (TD). We aim to generalize with the proposed approach to a wide range of scenarios; hence, we use the uniform distribution given in Table VII to train the NNs and in consecutive closed-loop evaluations. However, if the encountered SD is known better, we propose to include NNs trained on an a priori known SD and include it in the ensemble of NNs. In Fig. 10, the worse performance of an ensemble of REDS, purely trained on the uniform TD, followed by the soft-QP is shown when evaluated for samples taken from closed-loop simulations of the DEU_Cologne-63_5_I-1 scenario using the expert MIP-DM. In addition, Fig. 10 shows NNs trained on this SD, and how they can improve the prediction performance as single networks by ∼10% and in an ensemble by ∼3%.

C. Closed-Loop Results With SUMO Simulator

We compare the closed-loop performance of the REDS planner with varying numbers of NNs and of the expert

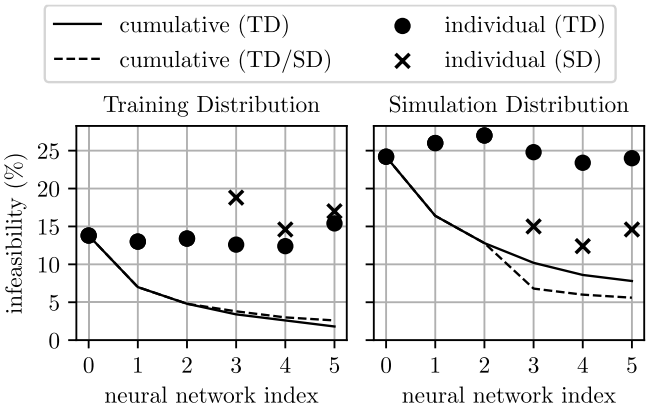

Fig. 10. Individual and cumulative REDS prediction performance for the TD and samples encountered by an SD. Two variants of the REDS ensemble are compared: one purely trained on the uniform TD and one ensemble with three NNs trained on the SD.

MIP-DM for the dense and sparse traffic in scenario DEU_Cologne-63_5_I-1, see Fig. 11. The closed-loop cost is computed by evaluating the objective (2) for the closed-loop trajectory and is separated into its components. The computation times are shown for the serial and parallel evaluation of the NNs. Similar to the open-loop evaluation in Fig. 6, the closed-loop results in Fig. 11 show a considerable performance gain when more NNs are added to the ensemble. Evaluating ten NNs in parallel within the REDS planner leads to nearly the same closed-loop performance as the expert MIP-DM. Remarkably, a parallel computation achieves a tremendous speed-up of the worst-case computation time of approximately 100 times. A serial evaluation of ten NNs could still be computed around 25 times faster for the maximum computation time compared with the expert MIP-DM. All computation times of the REDS planner variations are below the threshold of 200 ms, while the expert MIP-DM computation time exceeds the threshold in more than 50%, taking up to 4 s for one iteration. Table IV shows further the performance metrics. By using more NNs within an ensemble increases the average velocity and decreases the closed-loop cost toward the expert MIP-DM performance. Using six or ten NNs, a single situation occurred where a leading vehicle started to change lanes but halfway decided to change back toward the original lane, resulting in a collision, which in a real situation will be avoided by an emergency (braking) maneuver. Fig. 12 shows snapshots of a randomized closed-loop SUMO simulation of a dense

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

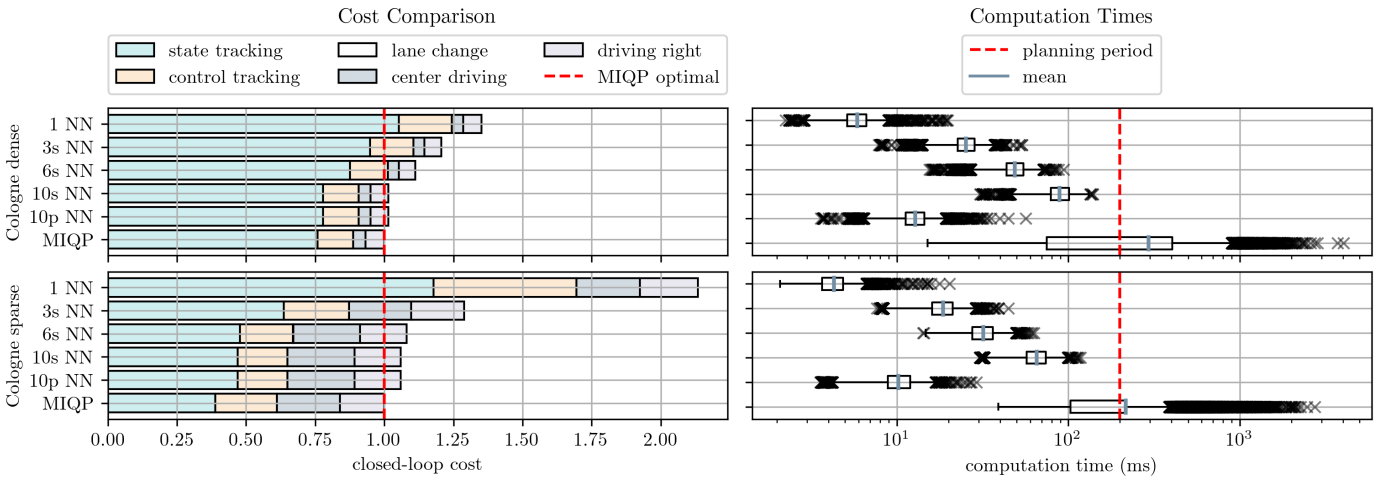

Fig. 11. Performance comparison for different numbers of serial (s) and parallel (p) NNs, for dense and sparse traffic in scenario `DEU_Cologne-63_5_I-1`. The closed-loop cost was computed by evaluating (2) for the closed-loop ego trajectory and normalized against the closed-loop cost of expert MIP-DM. The state tracking cost, including the lateral position and the desired velocity tracking error, contributes the most to the chosen weights.

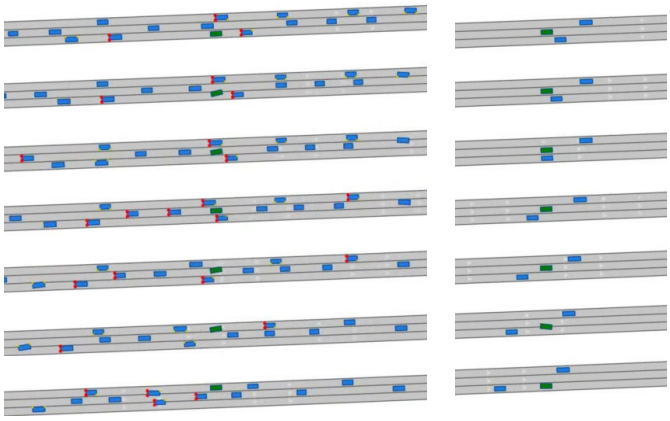

Fig. 12. Snapshots of simulated traffic scenario `DEU_Cologne-63_5_I-1` dense (left) and sparse (right) with `SUMO` and `CommonRoad`, showing the ego vehicle (green) and other vehicles (blue). The REDS planner is real-time feasible and used in combination with reference tracking NMPC, see Fig. 2. Red light at the rear of the vehicle indicates braking.

TABLE IV

CLOSED-LOOP EVALUATION FOR SCENARIO `DEU_Cologne-63_5_I-1` WITH DENSE AND SPARSE TRAFFIC

		`DEU_Cologne-63_5_I-1`-dense				
Property	Unit	MIQP	NN-1	NN-3	NN-6	NN-10
collisions		0	0	0	1	1
vel. mean	$\frac{m}{s}$	13.10	12.71	12.91	13.00	13.07
vel. min	$\frac{m}{s}$	0.41	1.17	0.00	0.00	0.00
lane changes		457	431	469	471	462
cost	$\frac{1}{s}$	49.48	66.89	59.68	55.02	50.22
		`DEU_Cologne-63_5_I-1`-sparse				
Property	Unit	MIQP	NN-1	NN-3	NN-6	NN-10
collisions		0	0	0	0	0
vel. mean	$\frac{m}{s}$	13.94	13.69	13.83	13.87	13.89
vel. min	$\frac{m}{s}$	7.10	7.10	7.10	7.10	7.10
lane changes		354	337	337	353	353
cost	$\frac{1}{s}$	5.81	12.40	7.49	6.27	6.16

and sparse traffic scenario `DEU_Cologne-63_5_I-1`. The *green* colored ego vehicle successfully plans lane changes and overtaking maneuvers to avoid collisions with other vehicles (blue), using the proposed REDS planner feeding a reference tracking NMPC.

X. CONCLUSION AND DISCUSSION

We proposed a supervised learning approach for achieving real-time feasibility for mixed-integer motion planning problems. Several concepts are introduced to achieve a nearly optimal closed-loop performance when compared with an expert MIQP planner. First, it was shown that inducing structural problem properties, such as invariance, equivariance, and recurrence into the NN architecture improves the prediction performance among other useful properties such as generalization to unseen data. Second, the soft-QP can correct wrong predictions, inevitably linked to the NN predictions and are able to evaluate an open-loop cost. This favors a parallel architecture of an ensemble of predictions and soft-QP computations to choose the lowest cost trajectory. In our experiments, adding NN to the ensemble improved the performance considerably and monotonously. This leads to the conclusion that NNs may be added as long as the computation time is below the planning threshold and as long as parallel resources are available. To further promote safety, an NLP, i.e., the FP, is used to plan a collision-free trajectory. The computational burden of the NLP is small, compared with the expert MIP-DM since it optimizes the trajectory only locally and therefore, omits the combinatorial variables.

Although we have evaluated the proposed approach for multilane traffic, the application to other urban driving scenarios is expected to perform similarly for a similar number of problem parameters due to the following. Many works, see [6], [29], [31], use similar MIQP formulations for a variety of automated driving (AD) scenarios, including traffic lights, blocked lanes, and merging. The formulations mainly differ in the specific environment. Many scenario specifics can be modeled by using obstacles and constraints related to the current lane [6], both considered in the presented approach. However, with an increasing number of problem parameters

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

TABLE V
PARAMETERS FOR CLOSED-LOOP SIMULATIONS IN SUMO

Category	Parameter	Value
General	episode length	30s
	nominal road velocity	$13.9\,\frac{\mathrm{m}}{\mathrm{s}}$
	maximum number of lanes	3
	lanes width d_{lane}	3.5m
	vehicle lengths	5.39m
	vehicle widths	2.07m
	ego desired velocity	$15\,\frac{\mathrm{m}}{\mathrm{s}}$
	maximum obstacle velocity	$23\,\frac{\mathrm{m}}{\mathrm{s}}$
	minimum obstacle velocity	$0.2\,\frac{\mathrm{m}}{\mathrm{s}}$
Dense scenario	traffic flow	$0.56\,\frac{\mathrm{vehicles}}{\mathrm{lane}\cdot\mathrm{s}}$
	traffic density	$0.04\,\frac{\mathrm{vehicles}}{\mathrm{lane}\cdot\mathrm{m}}$
Sparse scenario	traffic flow	$0.13\,\frac{\mathrm{vehicles}}{\mathrm{lane}\cdot\mathrm{s}}$
	traffic density	$0.01\,\frac{\mathrm{vehicles}}{\mathrm{lane}\cdot\mathrm{m}}$
FP	diag(Q)	$[1,1,1,1]^\top$
	diag(R)	$[1,1]^\top$
	slack weight w_{h}	10^6

TABLE VI
PARAMETERS IN MIQP FORMULATION (3) FOR EXPERT MIP-DM

Category	Parameter	Value
expert MIP-DM	diag(Q)	$[0,14,10,1]^\top$
	diag(R)	$[4,0.5]^\top$
	lane change weight w_{lc}	$3\cdot10^3$
	side preference weight w_{rght}	3
	safe distances $[\sigma_{\mathrm{f}},\sigma_{\mathrm{b}},\sigma_{\mathrm{l}},\sigma_{\mathrm{r}}]^\top$	$[0.5,12,0.5,0.5]^\top$m
	minimum controls $\underline{u}$	$[-10,-5]^\top\,\frac{\mathrm{m}}{\mathrm{s}}$
	maximum controls $\overline{u}$	$[3,5]^\top\,\frac{\mathrm{m}}{\mathrm{s}}$
	velocity ratio constraint α	0.3

TABLE VII
RANGES OF UNIFORM TRAINING DATA DISTRIBUTIONS

Par.	Range	Par.	Range
lat. pos.	$[0, n_{\mathrm{lanes}}d_{\mathrm{lanes}}] - \frac{d_{\mathrm{lanes}}}{2}$	lanes	[1,3]
obs. lon. pos.	[-120, 200]m	lon. vel.	$[0,30]\,\frac{\mathrm{m}}{\mathrm{s}}$
lat. vel.	$[-1,1]\,\frac{\mathrm{m}}{\mathrm{s}}$	obstacles	[1,5]

and rare events, the prediction of binary variables may become more challenging.

APPENDIX

In the following, we define the most important parameters used in the numerical simulations of this article. For the closed-loop simulations in SUMO, we used the parameter values in Table V. The reference velocity was set higher than the average nominal velocity to cause more overtaking maneuvers. For the optimization problems, we used the parameters shown in Table VI, in addition to the values presented in Table I. In Table VIII, NN hyperparameters are shown for architectures FF, LSTM, EDS, and REDS. The same LSTM layer is used for each set element (i.e., for each obstacle in our case) to achieve equivariance in the REDS network.

TABLE VIII
HYPERPARAMETERS FOR THE NN ARCHITECTURES

Category	Parameter	Value
General	activation function	ReLU
	batch size	128
	step size	$5\cdot10^{-5}$
	epochs	1500
	optimizer	adam
	loss function	cross-entropy
	weight decay	10^{-5}
	training samples	10^5
	test samples	10^3
FF	hidden layers	7
	neurons per layer	128
LSTM	layers	2
	hidden network size	128
	input network layers	2
	output network layers	2
EDS	layers	7
	equivariant hidden size	64
	unstructured hidden size	64
	input network layers	2
	input network hidden size	128
	output network layers	2
	output network hidden size	128
REDS	layers	7
	hidden network sizes	64
	input network layers	1
	output network layers	1
	eq. LSTM output network layers	1
	unstr. LSTM output network layers	1

REFERENCES

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

[2] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. 20th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1979, pp. 421–427.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[4] S. Di Cairano and I. V. Kolmanovsky, "Real-time optimization and model predictive control for aerospace and automotive applications," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 2392–2409.

[5] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annu. Rev. Control*, vol. 45, pp. 18–40, May 2018.

[6] R. Quirynen, S. Safaoui, and S. Di Cairano, "Real-time mixed-integer quadratic programming for vehicle decision making and motion planning," 2023, *arXiv:2308.10069*.

[7] A. Cauligi, P. Culbertson, E. Schmerling, M. Schwager, B. Stellato, and M. Pavone, "CoCo: Online mixed-integer control via supervised learning," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1447–1454, Apr. 2022.

[8] D. Bertsimas and B. Stellato, "The voice of optimization," *Mach. Learn.*, vol. 110, no. 2, pp. 249–277, Feb. 2021.

[9] R. Verschueren et al., "Acados—A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, no. 1, pp. 147–183, Oct. 2021.

[10] Q. Tran Dinh, S. Gumussoy, W. Michiels, and M. Diehl, "Combining convex–concave decompositions and linearization approaches for solving BMIs, with application to static output feedback," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1377–1390, Jun. 2012.

[11] Gurobi Optimization, LLC. (2023). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.gurobi.com

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

[12] A. Cauligi, A. Chakrabarty, S. D. Cairano, and R. Quirynen, "PRISM: Recurrent neural networks and presolve methods for fast mixed-integer optimal control," in *Proc. PMLR*, 2022, pp. 34–46.

[13] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[14] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, Aug. 2017, pp. 719–726.

[15] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, May 2020.

[16] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robot. Auto. Syst.*, vol. 174, Apr. 2024, Art. no. 104630.

[17] M. Sheckells, T. M. Caldwell, and M. Kobilarov, "Fast approximate path coordinate motion primitives for autonomous driving," in *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, Dec. 2017, pp. 837–842.

[18] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[19] O. Arslan, K. Berntorp, and P. Tsiotras, "Sampling-based algorithms for optimal motion planning using closed-loop prediction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4991–4996.

[20] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4523–4530.

[21] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 1386–1392.

[22] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 1954–1961.

[23] J. Li, X. Xie, Q. Lin, J. He, and J. M. Dolan, "Motion planning by search in derivative space and convex optimization with enlarged solution space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 13500–13507.

[24] S. Deolasee, Q. Lin, J. Li, and J. M. Dolan, "Spatio-temporal motion planning for autonomous vehicles with trapezoidal prism corridors and Bézier curves," in *Proc. Amer. Control Conf. (ACC)*, San Diego, CA, USA, May 2023, pp. 3207–3214.

[25] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, Jan. 2005.

[26] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1101–1115, Oct. 2015.

[27] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, 2016, pp. 205–210.

[28] K. Esterle, T. Kessler, and A. Knoll, "Optimal behavior planning for autonomous driving: A generic mixed-integer formulation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1914–1921.

[29] T. Kessler, K. Esterle, and A. Knoll, "Linear differential games for cooperative behavior planning of autonomous vehicles using mixed-integer programming," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 4060–4066.

[30] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A two-stage optimization-based motion planner for safe urban driving," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 822–834, Apr. 2022.

[31] T. Kessler, K. Esterle, and A. Knoll, "Mixed-integer motion planning on German roads within the Apollo driving stack," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 851–867, Jan. 2023.

[32] R. Reiter, M. Kirchengast, D. Watzenig, and M. Diehl, "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 99–106, 2021.

[33] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *Proc. 18th Eur. Control Conf. (ECC)*, Aug. 2019, pp. 1494–1499.

[34] V. Nair et al., "Solving mixed integer programs using neural networks," 2020, *arXiv:2012.13349*.

[35] E. B. Khalil, C. Morris, and A. Lodi, "MIP-GNN: A data-driven framework for guiding combinatorial solvers," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 9, pp. 10219–10227, doi: 10.1609/aaai.v36i9.21262.

[36] L. Russo, S. H. Nair, L. Glielmo, and F. Borrelli, "Learning for online mixed-integer model predictive control with parametric optimality certificates," *IEEE Control Syst. Lett.*, vol. 7, pp. 2215–2220, 2023.

[37] D. Bertsimas and B. Stellato, "Online mixed-integer optimization in milliseconds," *INFORMS J. Comput.*, vol. 34, no. 4, pp. 2229–2248, 2022.

[38] C. Xi, T. Shi, Y. Wu, and L. Sun, "Efficient motion planning for automated lane change based on imitation learning and mixed-integer optimization," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

[39] M. Srinivasan, A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. Di Cairano, "Fast multi-robot motion planning via imitation learning of mixed-integer programs," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 598–604, 2021.

[40] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction," *Neurocomputing*, vol. 445, pp. 298–308, Jul. 2021.

[41] E. Jo, M. Sunwoo, and M. Lee, "Vehicle trajectory prediction using hierarchical graph neural network for considering interaction among multimodal maneuvers," *Sensors*, vol. 21, no. 16, p. 5354, Aug. 2021.

[42] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker, "Dynamic input for deep reinforcement learning in autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7566–7573.

[43] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[44] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 5, no. 1, pp. 411–444, May 2022.

[45] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 384–406, Apr. 2018.

[46] T. Ersal et al., "Connected and automated road vehicles: State of the art and future challenges," *Vehicle Syst. Dyn.*, vol. 58, no. 5, pp. 672–704, Mar. 2020.

[47] K. Berntorp, A. Weiss, and S. D. Cairano, "Integer ambiguity resolution by mixture Kalman filter for improved GNSS precision," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 4, pp. 3170–3181, Aug. 2020.

[48] M. Greiff, S. Di Cairano, K. J. Kim, and K. Berntorp, "A system-level cooperative multiagent GNSS positioning solution," *IEEE Trans. Control Syst. Technol.*, vol. 32, no. 1, pp. 158–173, Jan. 2024.

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30. Red Hook, NY, USA: Curran Associates, 2017, pp. 1–8.

[50] R. Reiter and M. Diehl, "Parameterization approach of the frenet transformation for model predictive control of autonomous vehicles," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 2414–2419.

[51] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[52] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[53] P. Sollich and A. Krogh, "Learning with ensembles: How overfitting can be useful," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 8. Cambridge, MA, USA: MIT Press, 1995, pp. 1–11.

[54] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[55] F. Debrouwere et al., "Time-optimal path following for robots with convex–concave constraints using sequential convex programming," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013.

[56] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020.

[57] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[58] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.

[59] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.
[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–7.
[61] S. Krauss, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. thesis, Fac. Math. Natural Sci., Univ. Cologne, Cologne, Germany, Apr. 1998.
[62] J. Erdmann, "SUMO's lane-changing model," in *Proc. 2nd SUMO Conf. Modelling Mobility Open Data*, Berlin, Germany. Cham, Switzerland: Springer, May 2015, pp. 105–123.

Rudolf Reiter received the master's degree in electrical engineering from Graz University of Technology, Graz, Austria, in 2016, with a focus on control systems. He is currently pursuing the Ph.D. degree with the Marie-Skłodowska Curie Innovative Training Network position, University of Freiburg, Germany, under the supervision of Prof. Dr. Mortiz Diehl.

From 2016 to 2018, he worked as a Control Systems Specialist at the Anton Paar GmbH, Graz. From 2018 to 2021, he worked as a Researcher at the Virtual Vehicle Research Center, Graz. His research focus on learning- and optimization-based motion planning and control for autonomous vehicles.

Mr. Reiter is an Active Member of the Autonomous Racing Graz Team.

Rien Quirynen received the bachelor's degree in computer science and electrical engineering, the master's degree in mathematical engineering from KU Leuven, Leuven, Belgium, in 2010 and 2012, respectively, and the joint Ph.D. degree from KU Leuven, and the University of Freiburg, Freiburg im Breisgau, Germany, in 2016.

He worked as a Senior Research Scientist at the Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, from early 2017 until late 2023. He is currently a Staff Software Engineer at Stack AV, Pittsburgh, PA, USA. He has authored or coauthored more than 75 peer-reviewed papers in journals and conference proceedings and 25 patents. His research focuses on numerical optimization algorithms for decision making, motion planning, and predictive control of autonomous systems.

Dr. Quirynen received a four-year Ph.D. Scholarship from the Research Foundation-Flanders (FWO) from 2012 to 2016. He serves as an Associate Editor for the journal *Optimal Control Applications and Methods* (Wiley) and for the IEEE CSS Technology Conference Editorial Board.

Moritz Diehl received the diploma degree in physics and mathematics from Heidelberg University, Heidelberg, Germany, in 1999, and the Ph.D. degree in optimization and nonlinear model predictive control from the Interdisciplinary Center for Scientific Computing, Heidelberg University, in 2001.

From 2006 to 2013, he was a Professor with the Department of Electrical Engineering, KU Leuven University, Leuven, Belgium. Since 2013, he has been a Professor with the University of Freiburg, Freiburg im Breisgau, Germany, where he is currently the Head of the Systems Control and Optimization Laboratory, Department of Microsystems Engineering (IMTEK), and also with the Department of Mathematics. His research interests include optimization and control, spanning from numerical method development to applications in different branches of engineering, with a focus on embedded and renewable energy systems.

Stefano Di Cairano (Senior Member, IEEE) received the master's (Laurea) and Ph.D. degrees in information engineering from the University of Siena, Siena, Italy, in 2004 and 2008, respectively.

From 2008 to 2011, he was with Powertrain Control R&A, Ford Research and Advanced Engineering, Dearborn, MI, USA. Since 2011, he has been with the Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he is currently a Deputy Director and a Distinguished Research Scientist. His research focuses on optimization-based control and decision-making strategies for complex mechatronic systems, in automotive, factory automation, transportation systems, and aerospace. He has authored or coauthored more than 200 peer-reviewed papers in journals and conference proceedings and 80 patents. His research focuses on optimization-based control and decision-making strategies for complex mechatronic systems, in automotive, factory automation, transportation systems, and aerospace.

Dr. Di Cairano was the Chair of the IEEE CSS Technical Committee on Automotive Controls and the IEEE CSS Standing Committee on Standards. He is the Inaugural Chair of the IEEE CSS Technology Conference Editorial Board and was an Associate Editor of IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:48 UTC from IEEE Xplore. Restrictions apply.