IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 April 2026, accepted 7 June 2026, date of publication 10 June 2026, date of current version 15 June 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3702271

RESEARCH ARTICLE

A Comparison Benchmark for Distributed Hybrid MPC Control Methods: Distributed Vehicle Platooning

SAMUEL MALLICK, AZITA DABIRI, AND BART DE SCHUTTER, (Fellow, IEEE)
Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands

Corresponding author: Samuel Mallick (s.h.mallick@tudelft.nl)

This work was supported by European Research Council (ERC) under European Union's Horizon 2020 Research and Innovation Programme through CLariNet under Grant 101018826.

ABSTRACT Distributed model predictive control (MPC) is currently being investigated as a solution to the important control challenge presented by networks of hybrid dynamical systems, for which the computational complexity is often prohibitive. However, a benchmark problem for comparing distributed hybrid MPC solutions is absent from the literature. We propose distributed control of a platoon of autonomous vehicles as a comparison benchmark problem. The problem provides a complex and adaptable case study, upon which existing and future approaches to distributed MPC for hybrid systems can be evaluated. Two hybrid modeling frameworks are presented for the vehicle dynamics. Five hybrid MPC controllers are then evaluated and extensively assessed on the fleet of vehicles. Finally, we comment on the need for new efficient and high performing distributed MPC schemes for hybrid systems.

INDEX TERMS Autonomous vehicles, distributed model predictive control, hybrid systems, mixed integer optimization, piecewise affine systems.

I. INTRODUCTION

Distributed control of hybrid networks is an open problem in the field of systems and control [1]. Hybrid systems are dynamical systems that combine both continuous and discrete dynamics. Networks of hybrid systems represent many of the critical infrastructure systems in our society, such as transportation [2], energy [3], and water networks [4]. The grand societal challenge of sustainability demands the development of efficient and high-performance control approaches for these systems. Furthermore, these control methods must be safe and reliable, as these systems are safety critical.

Model predictive control (MPC) is an optimization-based control paradigm that has been highly successful in the area of distributed control for complex systems [5], [6]. MPC naturally handles multi-input-multi-output systems with constraints on the states and inputs, and is supported by a substantial body of literature on its stability and performance [7]. Distributed MPC replaces a global MPC controller with local MPC controllers for each subsystem. Distributed control is then carried out through some combination of solving the local MPC optimization problems, and communication between subsystems. Distributed MPC for linear systems with convex objective functions can be achieved with zero loss in performance via distributed optimization, as the global optimization problem is convex and can be solved to optimality distributively [8]. In contrast, distributed MPC of large-scale hybrid networks is a more complex control challenge. Compared to the linear case, the local optimization problems are highly nonlinear and non-convex, due to the combination of continuous and discrete dynamics. Consequently, distributed optimization techniques cannot in general guarantee the reconstruction of a globally optimal control solution. Furthermore, the combinatorial nature of optimizing the discrete components results in an exponential worst-case

The associate editor coordinating the review of this manuscript and approving it for publication was Inam Nutkani.

computational complexity for the optimization problems in practice, in contrast to the polynomial complexity of convex problems.

A prominent approach to deploy MPC for several relevant classes of hybrid systems is to transform the hybrid model into mixed-logical-dynamical (MLD) form [9]. The resulting MPC optimization problem is a mixed-integer linear/quadratic program (MILP/MIQP), for which mature solvers, e.g., Gurobi [10] and CPLEX [11], exist. The extension to distributed MPC of hybrid systems is then to find a control solution via subsystems solving local mixed-integer programs, and cooperating by communicating the solutions between neighboring subsystems. The local solutions can be found in parallel [12], iteratively solving and communicating to improve the global solution, or sequentially [13], [14], where local problems are solved one after the other, with local solutions communicated down the sequence. Another approach is to employ heuristic ways of choosing values for the integer variables while using distributed MPC to solve for values of the continuous variables [3], [15]. Alternatively, in [16] and [17] distributed MPC approaches for piecewise affine (PWA) systems, a class of hybrid systems, are proposed where the couplings between subsystems are treated as disturbances. The approaches are limited to regulation problems in which a robust terminal set exists.

An alternative set of approaches are based on distributed mixed-integer optimization, which enables distributed hybrid MPC by solving a global mixed-integer hybrid MPC problem distributively among subsystems. In [18] a distributed solution for large-scale MILPs is proposed using a dual formulation of the problem, guaranteeing a primally feasible solution and bounded suboptimality. This idea was then extended in [19] to include finite-time feasibility. In [20] a distributed large-scale MILP approach is proposed based on primal decomposition. These works, however, consider only inequality constraint coupling, and assume that the number of subsystems far exceeds the number of coupling constraints. These approaches are hence unsuitable for control problems with coupling in the cost or dynamics as, while cost and dynamics coupling can be converted to constraint coupling via introducing auxiliary variables [8], the number of constraints then far exceeds the number of subsystems. Finally, while the alternating direction method of multipliers (ADMM) is only guaranteed to converge for convex problems [21], it has been used as a distributed solution approach for mixed-integer programs. In [22] a modified ADMM procedure is presented for mixed-integer problems; however, the coupling between subproblems is restricted to only the binary variables. Additionally, in the context of distributed hybrid MPC, ADMM has been used as a heuristic method for solving the global optimization problem distributively without guarantees [2], [23].

Each of the methods referenced above introduces suboptimality relative to a centralized MPC controller, with no guarantees on its magnitude. As the approaches are demonstrated on a range of different case studies, with the nature and complexity of the control problem varying from example to example, it is difficult to analyze the relative performance of each controller; a benchmark problem is required. To the best of the authors' knowledge, a standardized benchmark problem, based on a relevant real-world challenge, for evaluating the effectiveness of distributed hybrid MPC approaches is missing from the literature. The goal of this paper is to propose such a benchmark.

We introduce the platooning of a fleet of vehicles, where vehicle gear shifting is explicitly optimized, as the benchmark problem. Control of a single SMART[1] vehicle has been proposed as a comparative benchmark problem for centralized MPC of hybrid systems [25]. Vehicle platooning in the context of distributed control has been explored using non-hybrid models in [26], [27], and [28]. In these works the inherent hybrid nature of discrete gear choices is either assumed to be absent or is neglected. While vehicles with continuous-variable-transmission (CVT) exist, where there is no notion of a discrete gear, these account for a very small share of vehicles only [29]. Moreover, explicit gear management has been identified as important for fuel consumption, effecting the efficiency of the engine, and for tracking and platooning behavior, as gear shifts can introduce large deviations in speed and inter-vehicular distance [30]. For control of a single autonomous vehicle, optimization of both continuous control inputs and discrete gear choices has been addressed in [31], [32], and [33]. However, these approaches do not extend to platoons of vehicles, and they are inherently centralized control approaches. The only previous work to consider control of gear shifts in platooning is [30], where the gear shifting control is decoupled from the longitudinal vehicle control. Finally, in [34] a hybrid distributed MPC approach is formulated for platoon control; however, the hybrid nature of the problem comes from lane changes, while the vehicles have non-hybrid dynamics.

The current paper makes the following contributions to the state of the art:

- A platooning problem is proposed as a useful comparison benchmark problem for distributed hybrid MPC approaches. The problem scrutinizes, for distributed hybrid MPC strategies, the handling of many discrete decision variables, through the (predicted) gear transitions, and the effective coordination and communication between distributed controllers, with heavy coupling between controllers arising from position tracking and coupled safety constraints.
- Two hybrid models are introduced for the vehicles with explicit gear management. The platooning problem for hybrid vehicles with gear transitions is formulated, with a variety of *tuning knobs* determining the complexity of the control challenge.
- An open-source code base for the benchmark problem is provided, where alternative models, controllers, and tasks can be evaluated.

[1] SMART is a German vehicle manufacturer [24].

- As a first use of the benchmark, five representative existing control strategies are extensively evaluated and their performance, complexity, and characteristics are discussed.

This paper is organized as follows. Section II introduces two hybrid models for vehicles with explicit gear management. Section III formulates the benchmark control problem. Section IV gives a brief overview of each of the control methods to be evaluated. Section V provides simulation results, and compares and assesses the behaviors of the controllers. Finally, in Section VI we provide concluding remarks and discuss the open challenges highlighted by the benchmark case study.

II. VEHICLE MODELS

In this section we detail two modeling approaches for the dynamics of a vehicle with explicit gear management. These models directly use the friction model from [25]; however, the presented gear models are novel. Whereas [25] introduces errors into the gear traction values, in order to express the traction as an affine function of the gear, in this work we present two models that use the true traction values.

A. MODELS

Considering a vehicle driving forwards, an accurate model of its dynamics is

$$m\ddot{s}(t) + c\dot{s}^2 + \mu mg = b(j, \dot{s})u(t), \tag{1}$$

where $s(t)$ is the position at time t, $j \in \{1, \ldots, 6\}$ is the selected gear, and $b(j, \dot{s})u(t)$ is a traction force that is proportional to the normalized throttle position $u(t)$. In (1) g is gravitational acceleration, μ is the Coulomb friction coefficient, c is the viscous friction coefficient, and m is the vehicle mass (values in Table 2). Defining the state as position and velocity via $x = \begin{bmatrix} s & \dot{s} \end{bmatrix}^\top$, the dynamics are expressed as

$$\dot{x} = A(x) + B(j, x)u \tag{2}$$

where

$$A(x) = \begin{bmatrix} x_2 \\ -(1/m)f(x_2) - \mu g \end{bmatrix}, \; B(j, x) = \begin{bmatrix} 0 \\ b(j, x_2)/m \end{bmatrix}, \tag{3}$$

and with $f(x_2) = cx_2^2$. The dynamics are nonlinear, due to the quadratic friction, and hybrid, due to the discrete variable j. The quadratic friction $f(x_2)$ is depicted in Figure 1 and the traction force $b(j, x_2)$ in Figure 2a. Table 1 gives the maximum traction value for each gear and the velocity ranges for which this maximum traction is constant.

A common approach to 'hybridize' a nonlinearity is to use a PWA approximation, replacing a nonlinear curve with a series of affine pieces. As in [25], we approximate f with the PWA function $\hat{f}$ in Figure 1, using two affine regions,

$$\hat{f}(x_2) = \begin{cases} a_1 x_2 + c_1 & x_2 \leq \alpha \\ a_2 x_2 + c_2 & x_2 > \alpha \end{cases}, \tag{4}$$

TABLE 1. Gear traction.

Gear j	$T_{ractionforce}$ $b(j)(N)$	$M_{in.vel.}$ (m/s)	$M_{ax.vel.}$ (m/s)
I	4057	3.94	9.46
II	2945	5.43	13.04
III	2116	7.56	18.15
IV	1607	9.96	23.90
V	1166	13.70	32.93
VI	838	19.10	45.84

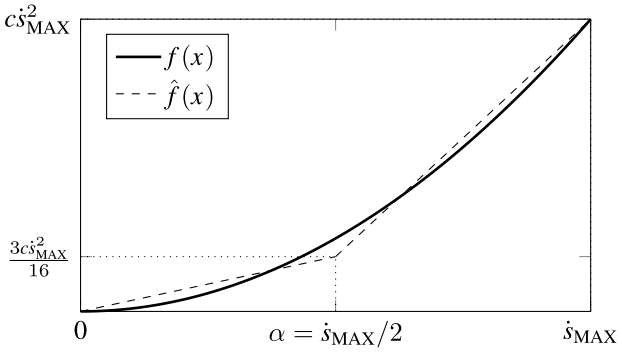


FIGURE 1. PWA friction approximation. True quadratic function (solid), piecewise approximation (dashed).

where $\alpha = \dot{s}_{\text{MAX}}/2$, and $\dot{s}_{\text{MAX}}$ is a maximum velocity, defined in Table 2. A PWA approximation of $A(x)$ is then

$$A_{\text{PWA}}(x) = \begin{bmatrix} x_2 \\ -\hat{f}(x_2)/m - \mu g \end{bmatrix}. \tag{5}$$

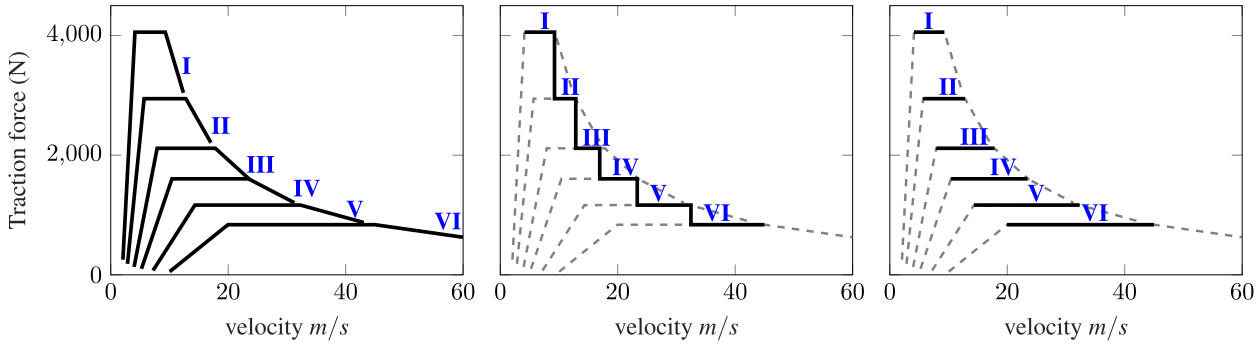
We explore two models for the hybrid gear dynamics $B(j, x)$. The first approach considers the gear choice as dependent on the velocity, giving a PWA model of $B(j, x)$. The second approach will consider the gear choice as an independent discrete decision variable.

1) PWA GEAR MODEL

For a PWA model of the gear dynamics, we consider the regions of the traction curves in Figure 2a where the traction is constant. The velocity is then partitioned, and a gear is chosen for each region, such that the velocity-gear space is artificially constrained to be a one-to-one mapping, without introducing any approximation in the vehicle dynamics. We take the mid-point of a gear's constant velocity range as the lower bound for the PWA region associated with that gear. The PWA gear model, depicted in Figure 2b, is then $B_{\text{PWA}}(x) = \begin{bmatrix} 0 & \hat{b}(x_2)/m \end{bmatrix}^\top$ where

$$\hat{b}(x_2) = \begin{cases} b_{1,\text{H}} & v_{1,\text{L}} \leq x_2 < \dfrac{v_{2,\text{L}} + v_{2,\text{H}}}{2} \\ b_{2,\text{H}} & \dfrac{v_{2,\text{L}} + v_{2,\text{H}}}{2} \leq x_2 < \dfrac{v_{3,\text{L}} + v_{3,\text{H}}}{2} \\ \vdots & \\ b_{6,\text{H}} & \dfrac{v_{6,\text{L}} + v_{6,\text{H}}}{2} \leq x_2 < v_{6,\text{H}} \end{cases}, \tag{6}$$

and $b_{j,\text{H}}$, $v_{j,\text{L}}$, and $v_{j,\text{H}}$, are the maximum traction, minimum velocity, and maximum velocity bounds for gear j, as given in Table 1. The PWA gear model no longer includes a discrete



(a) Full $b(j, x_2)$ (b) PWA $\hat{b}(x_2)$ (c) Discrete input $b(\delta_1, \ldots, \delta_6)$

FIGURE 2. Gear models.

decision variable j, and is a function only of the state x. Combining $B_{\text{PWA}}(x)$ with $A_{\text{PWA}}(x)$ gives the PWA model

$$\dot{x} = A_{\text{PWA}}(x) + B_{\text{PWA}}(x)u \tag{7}$$

with seven affine regions determined by the velocity (six from the gears and an extra from the friction). We will refer to this model as Model I. This model can serve as the prediction model in an MPC controller for the vehicle, and can be converted into a variety of equivalent hybrid models [35], resulting in different MPC optimization problems, e.g., an MLD model, giving a mixed-integer MPC problem [9].

2) DISCRETE-INPUT GEAR MODEL

The second gear model we consider describes the gear choice as a discrete input. Again the traction curves are restricted to the regions of constant traction, and each gear is restricted to operate only in these regions. However, unlike the PWA approximation, the mapping from velocity to gear is not one-to-one. To represent the gear choice, six binary variables $\delta_1 - \delta_6$ are introduced, one for each gear. When δ_j is equal to one, gear j is chosen, with the constraint

$$\sum_{j=1}^{6} \delta_j = 1 \tag{8}$$

ensuring that only one gear is active. Figure 2c depicts the discrete-input gear model

$$b(\delta_1, \ldots, \delta_6) = \sum_{j=1}^{6} \delta_j b_{j,\text{H}}, \tag{9}$$

with the B matrix is then

$$B_{\text{DISC}}(\delta_1, \ldots, \delta_6) = \begin{bmatrix} 0 \\ b(\delta_1, \ldots, \delta_6)/m \end{bmatrix}. \tag{10}$$

The nonlinearity originating from the multiplication of binary and continuous decision variables in $B_{\text{DISC}}(\delta_1, \ldots, \delta_6)u$ can be reformulated as the linear expression $B_{\text{MLD}}(\delta_1, \ldots, \delta_6, u)$ through the addition of auxiliary variables and mixed-integer linear constraints (see [9] for details). Finally, restricting the gears to operate within the velocity regions of constant traction is achieved via the inclusion of additional mixed-integer constraints that encode the logical expressions, e.g.,

$$\big(\delta_j = 1 \implies x_2 \le v_{j,\text{H}}\big) \iff x_2 - v_{j,\text{H}} \le M_{\text{H}}(1 - \delta_j), \tag{11}$$

where $M_{\text{H}} = \max_x(x_2 - v_{1,\text{H}})$, and

$$\big(\delta_j = 1 \implies x_2 \ge v_{j,\text{L}}\big) \iff v_{j,\text{L}} - x_2 \le M_{\text{L}}(1 - \delta_j), \tag{12}$$

where $M_{\text{L}} = \max_x(v_{1,\text{L}} - x_2)$.

Converting $A_{\text{PWA}}(x)$ to MLD form (again, see [9] for details) $A_{\text{MLD}}(x)$ and combining with $B_{\text{MLD}}(\delta_1, \ldots, \delta_6, u)$ gives the MLD model

$$\begin{aligned} \dot{x} &= A_{\text{MLD}}(x) + B_{\text{MLD}}(\delta_1, \ldots, \delta_6, u) \\ \text{s.t.} &\quad \text{mixed-integer equations}, \end{aligned} \tag{13}$$

with eight binary variables (six from the gears and two from the PWA regions of the friction). We will refer to this model as Model II. This model can serve as the prediction model in an MPC controller, resulting in a mixed-integer optimization problem. However, as the velocity to gear mapping is not one-to-one, the model cannot be converted into PWA form.

In an MPC context a discrete-time model is required. Both models are discretized using the forward Euler method and a sampling time T. In the following, all references to models refer to the discrete-time versions, and k represents the discrete-time step counter.

III. BENCHMARK PROBLEM

Consider the control problem where a platoon of M vehicles track each other's position and velocity. Define the set of vehicles as $\mathcal{M} = \{1, \ldots, M\}$, with vehicle i having mass m_i, and state $x^{(i)}$ containing its position and velocity. Without loss of generality we assume the set is sorted by the position of the vehicles, i.e., for the first vehicle $i = 1$, and vehicle i is behind vehicle $i - 1$. Only longitudinal motion is considered, i.e., there is only one lane with no overtaking. One vehicle $l \in \mathcal{M}$, henceforth referred to as the leader, is provided with a reference trajectory $r = \{r(k)\}_{k \ge 0}$, where $r(k) \in \mathbb{R}^2$ is the desired state of the leader vehicle at

time step k. All other vehicles track the states of the preceding and succeeding vehicles, with the desired difference between the state of vehicle i and vehicle $i-1$ given by an inter-vehicle spacing policy $\eta(x^{(i)}) \in \mathbb{R}^2$. It is assumed that each vehicle can take error-free measurements of the current position and velocity of the preceding and succeeding vehicles.

Within the scope of this problem there a set of *tuning knobs* that alter the complexity of the control challenge:

- The first tuning knob is the number of vehicles M, with more vehicles presenting a larger global system, and more subsystems that must coordinate.
- The second tuning knob is the reference trajectory r, which determines the overall behavior of the platoon, and can trigger smooth or aggressive behaviors.
- The third tuning knob is the inter-vehicle spacing policy η. Constant-distance, $\eta(x^{(i)}) = \begin{bmatrix} d_0 & 0 \end{bmatrix}^\top$, and velocity-dependent, $\eta(x^{(i)}) = \begin{bmatrix} t_0 x_2^{(i)} + d_0 & 0 \end{bmatrix}^\top$, spacing policies are considered, where d_0 and t_0 are a fixed distance and time, respectively. A velocity-dependent spacing policy increases the effective coupling between vehicles, as vehicle $i-1$ maintains a spacing from vehicle i that depends on the velocity of vehicle i.
- The fourth tuning knob is vehicle inhomogeneity, changing the dynamics (2) through variation in the masses m_i. Vehicle inhomogeneity makes the control task more challenging and enhances the need for cooperation, e.g., lighter vehicles can accelerate faster, and without coordination will not be trackable by heavier vehicles.
- The fifth tuning knob is leader position. Platooning typically considers the front vehicle as the leader, simplifying the problem, as the front vehicle is not restricted by a preceding vehicle. By assigning the leader role to a vehicle within the platoon, $l \neq 1$, the reference tracking is more challenging.

Remark 1: We note that varying communication topologies have been explored in [26]. Additionally, predictive control under state measurement errors has been considered in [36], and non-ideal communication between vehicles has been addressed in the form of, e.g., quantization [37], communication delays [38], and asynchronous updates [39]. Finally, mixed-traffic scenarios including human drivers are considered in [40]. In building this benchmark case study, we allow the communication topology to vary with the controllers, with the level and type of communication being a point of comparison. We also assume no measurement errors, perfect communication, and no human drivers, to maintain the focus on nominal controller performance.

Remark 2: While stability of individual vehicles [27], [41] and string stability of the network [26] are important considerations in platooning, they fall outside the scope of the current work. Here, we focus specifically on establishing a benchmark for the performance of distributed hybrid MPC controllers, prioritizing tracking performance alongside communication and computation time requirements. Incorporating stability guarantees within this benchmark framework is left as a direction for future work.

A. CONSTRAINTS

The states and inputs of the vehicles are subject to constraints. Numerical values for the constraint coefficients used in our experiments are given in Table 2. Velocity and acceleration constraints are imposed for safety and comfort:

$$v_{\text{MIN}} \leq x_2^{(i)}(k) \leq v_{\text{MAX}} \tag{14a}$$

$$a_{\text{MIN}} T \leq x_2^{(i)}(k+1) - x_2^{(i)}(k) \leq a_{\text{MAX}} T, \tag{14b}$$

where $v_{\text{MIN}} > 0$, $v_{\text{MAX}} > 0$, $a_{\text{MIN}} < 0$, and $a_{\text{MAX}} > 0$ are velocity and acceleration bounds respectively. Note that the lower bound on velocity is to maintain the validity of the constant traction approximations in the models, and to ensure the vehicles drive forwards. The normalized throttle input must be constrained as

$$-u_{\text{MAX}} \leq u^{(i)}(k) \leq u_{\text{MAX}}. \tag{15}$$

The vehicles (excluding the front vehicle) must maintain a safe distance d_{safe} from the preceding vehicle:

$$x_1^{(i)}(k) \leq x_1^{(i-1)}(k) - d_{\text{safe}}. \tag{16}$$

Finally, we impose non-restrictive constraints on the position, as the PWA to MLD model conversion requires that the states are bounded [9]:

$$p_{\text{MIN}} \leq x_1^{(i)}(k) \leq p_{\text{MAX}}, \tag{17}$$

where p_{MIN} and p_{MAX} are position bounds. This is not restrictive as, in an MPC approach, the vehicles can always reset the origin of the position measurements.

B. OPTIMAL CONTROL PROBLEM

MPC generates control signal sequences $\{\mathbf{u}^{(i)}\}_{i\in\mathcal{M}} = \left\{\left(u^{(i)}(0), \ldots, u^{(i)}(N-1)\right)\right\}_{i\in\mathcal{M}}$ by solving an optimization problem that minimizes a cost function over the prediction horizon N, subject to the constraints (14)-(17), and the vehicle models ((7) or (13)). The first elements of these control sequences $\left\{u^{(i)}(0)\right\}_{i\in\mathcal{M}}$ are then applied, and the optimization is performed again at the next time step in a receding horizon fashion. Define the centralized MPC optimization problem as

$$\begin{aligned}
\min_{\{(\mathbf{u}^{(i)},\mathbf{x}^{(i)},\mathbf{j}^{(i)})\}_{i\in\mathcal{M}}} \quad & \sum_{k=0}^{N} \Big(\|x^{(l)}(k) - r(k)\|_{Q_x} \\
& + \sum_{i\in\mathcal{M}\setminus\{1\}} \|x^{(i)}(k) - x^{(i-1)}(k) - \eta\big(x^{(i)}(k)\big)\|_{Q_x} \Big) \\
& + \sum_{i\in\mathcal{M}} \sum_{k=0}^{N-1} \|u^{(i)}(k)\|_{Q_u} \\
\text{s.t. for } i \in \mathcal{M} \quad & \\
& (14), (16), (17), \ k = 1, \ldots, N \\
& \text{vehicle model, } (15), \ k = 0, \ldots, N-1 \\
& x^{(i)}(0) = x^{(i)}
\end{aligned} \tag{18}$$

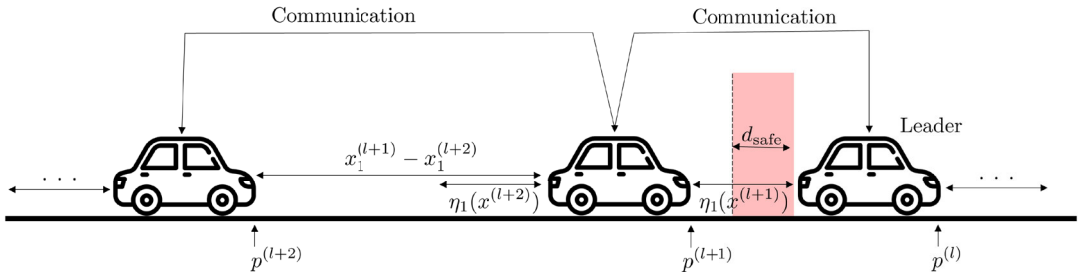


FIGURE 3. Platoon control.

where the bold $\mathbf{u}/\mathbf{x}/\mathbf{j}$ represent the control, state, and gear decision variables over the prediction horizon, Q_x and Q_u are weight matrices, and $x^{(i)}$ is the current state of vehicle i. The operator $\|\cdot\|$ is a general norm penalty, which will be specified in Section V.

Remark 3: The gear decision variables $\mathbf{j}$ enter (18) through the vehicle model, either explicitly as a discrete input in Model II, or implicitly, dependent on the velocity, in Model I. A possible augmentation of (18) would penalize gear changes explicitly in the cost to reduce fuel consumption [31].

The optimization problem (18) has coupling between the vehicles in the cost function and in the constraints. For distributed MPC, the centralized MPC problem is broken into local subproblems for each vehicle $i \in \mathcal{M}$:

$$\begin{aligned}
\min_{\mathbf{u}^{(i)},\mathbf{x}^{(i)},\mathbf{j}^{(i)}} \; & \sum_{k=0}^{N} \|x^{(i)}(k) - \bar{x}^{(i-1)}(k) - \eta\big(x^{(i)}(k)\big)\|_{Q_x} \\
& + \|\bar{x}^{(i+1)}(k) - x^{(i)}(k) - \eta\big(\bar{x}^{(i+1)}(k)\big)\|_{Q_x} \\
& + \sum_{k=0}^{N-1} \|u^{(i)}(k)\|_{Q_u} \\
\text{s.t. } & : (14), (17), \; k = 1, \ldots, N \\
& x_1^{(i)}(k) \le \bar{x}_1^{(i-1)}(k) - d_{\text{safe}}, \; k = 1, \ldots, N \\
& \bar{x}_1^{(i+1)}(k) \le x_1^{(i)}(k) - d_{\text{safe}}, \; k = 1, \ldots, N \\
& \text{vehicle model, } (15), \; k = 0, \ldots, N-1 \\
& x^{(i)}(0) = x^{(i)}
\end{aligned} \tag{19}$$

where the new variables $\bar{\mathbf{x}}^{(i-1)}$ and $\bar{\mathbf{x}}^{(i+1)}$ represent some approximation or assumption on the trajectory of the vehicles in front of and behind vehicle i. The key consideration in distributed MPC approaches is then how vehicles decide or agree upon $\bar{\mathbf{x}}^{(i-1)}$ and $\bar{\mathbf{x}}^{(i+1)}$.

Naturally, the local subproblem for the leader uses the reference trajectory r in the cost. Likewise, the local subproblems for the front and rear vehicles will not include cost terms or safety constraints for a preceding or succeeding vehicle, respectively. For brevity we do not present these subproblems explicitly.

C. FEASIBILITY

We briefly discuss the feasibility of the optimization problems (18) and (19), considering each of the constraints introduced in Section III-A. The position constraints (17) are only required to give a bounded state space; hence, p_{MIN} and p_{MAX} can be chosen arbitrarily small and large, respectively, such that the position constraints are always feasible. The input constraints (15) can always be satisfied as the control inputs $u^{(i)}(k)$ are independent decision variables. The safe distance constraints (16) are coupled between adjacent vehicles, and are hence handled differently by different distributed controllers. Consequently, these constraints are softened with slack variables that are penalized in the cost, such that the softened safe distance constraints can always be satisfied by making the slack variables non-zero. The number of times that two vehicles breach the safe distance then becomes a comparison point between different distributed controllers.

The velocity and acceleration constraints (14) require closer attention. For brevity we describe an $x_2(k)$ value such that $v_{\text{MIN}} \le x_2(k) \le v_{\text{MAX}}$ as a 'viable' $x_2(k)$. Note that for all viable $x_2(k)$, the existence of a control input $u(k)$ such that $x_2(k+1) = x_2(k)$ is sufficient to ensure the feasibility of both the velocity and acceleration constraints (14) for time steps after k. We then provide a sufficient condition on the vehicle mass m such that this control input exists. For the PWA model, i.e., Model I, the velocity dynamics are

$$\begin{aligned}
x_2(k+1) = x_2(k) + T\Big(& -\frac{1}{m}\hat{f}\big(x_2(k)\big) \\
& - \mu g + \hat{b}\big(x_2(k)\big)u(k)\Big).
\end{aligned} \tag{20}$$

Solving for $x_2(k+1) = x_2(k)$ gives

$$u(k) = \big(\frac{1}{m}\hat{f}\big(x_2(k)\big) + \mu g\big)/\hat{b}\big(x_2(k)\big), \tag{21}$$

with $u(k) > 0$ as all terms on the right-hand side are positive. We must then check only that $u(k)$ in (21) is less than u_{MAX}, i.e., for all viable x_2

$$m \ge \frac{\hat{f}(x_2)}{\hat{b}(x_2)u_{\text{MAX}} - \mu g}. \tag{22}$$

As $\hat{b}$ is constant within each PWA region and $\hat{f}$ is monotonically increasing with x_2 (see Figure 1), it is sufficient to evaluate the bound (22) at the maximum x_2 value within each PWA region, taking the largest bound as the requirement on m. Following the same steps for the discrete-input model,

Model II, it is sufficient to bound m for every valid gear for a given x_2, i.e., for viable x_2, and for all j such that $v_{j,\mathrm{L}} \leq x_2 \leq v_{j,\mathrm{H}}$,

$$m \geq \frac{\hat{f}(x_2)}{b(j, x_2)u_{\mathrm{MAX}} - \mu g}. \tag{23}$$

By the same arguments as above, it is sufficient to evaluate the bound (23) for the maximum x_2 of each gear, i.e., $x_2 = v_{j,\mathrm{H}}$, and within each friction PWA region, i.e., $x_2 = \alpha$ and $x_2 = \dot{s}_{\mathrm{MAX}}$, taking the largest bound for m. For the vehicle parameters listed in this paper (see Section V), this gives $m \geq 2.82$ kg for both models, such that for vehicle masses $m \geq 2.82$ kg, and viable $x_2(k)$, the constraints (14) are feasible for time steps after k. The final consideration to make is that, due to model errors between the prediction models and the real vehicle models, $x_2(0)$ could be outside the viable range, in which case a bound on the model error is required to determine the existence of a $u(0)$ such that $x_2(1)$ is viable. Alternatively, using a bounded model error, robust MPC techniques [42] such as constraint tightening could be applied to ensure $x_2(0)$ is always viable. Quantifying the model mismatch is beyond the scope of this paper, and we instead assume that the horizon is long enough and the sample time is small enough such that $x_2(0)$ remains in the viable range, such that the velocity and acceleration constraints (14) are feasible for all k.

D. PERFORMANCE MEASURES

To compare the performance of different controllers on the benchmark problem, consider the tracking performance

$$\begin{aligned} J = \sum_{k=0}^{K_{\mathrm{SIM}}} \Bigg(& \|x^{(l)}(k) - r(k)\|_{Q_x} \\ & + \sum_{i \in \mathcal{M} \setminus \{1\}} \|x^{(i)}(k) - x^{(i-1)}(k) - \eta\big(x^{(i)}(k)\big)\|_{Q_x} \\ & + \sum_{i \in \mathcal{M}} \|u^{(i)}(k)\|_{Q_u} \Bigg), \end{aligned} \tag{24}$$

where K_{SIM} is the length of a simulation. This measure evaluates how well the vehicles track each other, through penalizing the state differences, and how efficiently, through penalizing the throttle input. Additionally, the number of times the safe distance d_{safe} between vehicles is breached can be considered, indicating when there is insufficient agreement between vehicles on the values of coupled states. In Section V we also evaluate the properties of the controllers that influence their applicability: computation time, amount of communication between vehicles, and the local memory required for solving the optimization problems.

E. OPEN SOURCE CODE

An open source code base for the benchmark problem is available at https://github.com/SamuelMallick/hybrid-vehicle-platoon.git under the GNU General Public license. The code base includes the simulation environment, the two vehicle models presented in this paper, and implementations of all controllers compared in this paper, such that all algorithms and simulations in this work can be reproduced, and the benchmark can be extended. The tuning knobs of the control tasks, e.g., the number of vehicles in the platoon, can be easily modified via global parameters. The current controllers use the Gurobi [10] optimizer to solve the MPC optimal control problems. Alternative vehicle models and control strategies can be added in a plug-and-play manner.

IV. HYBRID MPC CONTROLLERS

In this section we give a high-level introduction of the five hybrid MPC controllers that are evaluated on the benchmark problem. For exact implementation details, we refer to the open-source code base associated with this work, and the citations in the following. These are:

- Centralized MPC
- Decentralized MPC
- Sequential distributed MPC
- Event-based distributed MPC
- ADMM-based distributed MPC.

For each controller, both Model I and II can be used as the prediction model. When Model I is used, it is converted into MLD form. Hence, all MPC optimization problems in the following are MILPs or MIQPs, and can be solved with a branch-and-bound (BnB) solver. BnB is a widely used methodology for solving combinatorial optimization problems, involving enumerating possible solutions to the problem and storing partial solutions in a tree structure. The tree structure branches, creating subproblems, by partitioning the solution space, while bounds on the optimum can be used to prune entire parts of the tree, where the solutions are known to be suboptimal. In mixed-integer optimization the subproblems are created by relaxing integer variables to be continuous. Branching is then done by partitioning the continuous feasible region. For an extended overview of BnB, in particular for mixed-integer programming, see [43].

A. CENTRALIZED

The centralized MPC controller solves (18) directly at each time step, finding all control inputs for all vehicles in a single optimization problem. Using a BnB solver, the global optimum of (18) can be found. As such, this controller serves as the baseline performance for the following distributed controllers.

B. DECENTRALIZED MPC

The decentralized MPC controller finds control inputs by solving the local subproblems (19) in parallel, without any communication between the vehicles. Decentralized MPC has been explored for linear systems [44], and can trivially be extended to hybrid systems, with the only difference being the prediction model in the local MPC problems. While traditional decentralized MPC *ignores* coupling,

for the platoon control case we take advantage of a vehicle's ability to measure, in an error-free way, the state of adjacent vehicles. At each time step vehicles measure the position and velocity of adjacent vehicles and, assuming constant velocities,[2] extrapolate the positions to generate the estimates $\bar{\mathbf{x}}^{(i-1)}$ and $\bar{\mathbf{x}}^{(i+1)}$.

C. SEQUENTIAL DISTRIBUTED MPC

Sequential distributed MPC approaches solve the local subproblems serially in a fixed sequence. After a subsystem solves its subproblem, it communicates the solution to the next subsystems in the sequence. This idea has been implemented for linear systems in [13], and hybrid systems in [14].

For a platoon, the natural sequence of subproblems is the leader first, followed by the adjacent vehicles, i.e., vehicle l first, then vehicles $l-1$ and $l+1$, then vehicles $l-2$ and $l+2$, and so on until all vehicles have solved their subproblems. When vehicle i solves its subproblem, it communicates its predicted state to vehicle $i-1$ and $i+1$. Therefore, vehicles have perfect knowledge of $\bar{\mathbf{x}}^{(i-1)}$ or $\bar{\mathbf{x}}^{(i+1)}$, when solving (19), if vehicles $i-1$ or $i+1$ were earlier in the sequence than vehicle i. For $\bar{\mathbf{x}}^{(i-1)}$ or $\bar{\mathbf{x}}^{(i+1)}$ of vehicles $i-1$ or $i+1$ after vehicle i in the sequence, vehicle i uses the solutions for $\bar{\mathbf{x}}^{(i-1)}$ or $\bar{\mathbf{x}}^{(i+1)}$ from the previous time step, shifted by one time step, assuming constant velocity for the shifting.

D. EVENT-BASED DISTRIBUTED MPC

Event-based distributed MPC [12] involves each subsystem solving local problems in parallel, and communicating the solutions only in the event of a significant cost improvement. The local problems are enlarged subproblems that consider the control inputs and states of coupled subsystems as decision variables. In the platoon, coupled subsystems are adjacent vehicles, and each vehicle solves a subproblem considering the states and inputs of the preceding and succeeding vehicles as additional decision variables. Starting from some base solution, all vehicles solve their subproblems in parallel, and compute a cost-improvement from the base solution. If a cost-improvement threshold is achieved, the vehicle with the highest cost improvement communicates its optimized trajectories to other vehicles, that then serve as the new base solutions, and the process is repeated. In this way the method uses 'parallel computation and serial communication' to achieve agreement and improvement on the shared variables. The base solutions at each time step are the shifted solutions from the previous time step, again extrapolating positions assuming constant velocity. For the initial time step of a simulation, when no previous solutions are available, vehicles use measurement to generate a base solution, as in the decentralized approach.

Originally, the approach would continue to iterate computation and event-based communication until the sampling time is exhausted [12]. In this comparison, where the computation time is not restricted, we instead run the algorithm for a range of fixed numbers of iterations.

As vehicles optimize over the trajectories of adjacent vehicles, this approach assumes that vehicles have exact knowledge of the dynamics of their adjacent vehicles. This may not always be a reasonable assumption, particularly in the case of heterogeneous platoons.

E. ADMM-BASED DISTRIBUTED MPC

While ADMM is only guaranteed to converge for convex optimization problems, in the literature it has been applied to hybrid, non-convex, control problems 'naively', in the hope that, while not guaranteed to converge, the resulting solution will be close to the true optimum [2]. An ADMM approach involves augmenting the local cost function in (19) with penalty terms that penalize the difference between the assumed coupled variables $\bar{\mathbf{x}}^{(i-1)}$ and $\bar{\mathbf{x}}^{(i+1)}$ and their true values. Vehicles iteratively solve their subproblems and communicate the solutions to adjacent vehicles. In the convex case, at convergence, the coupled variables will be agreed upon across the network, and the local optimizers will converge to the optimizers of the centralized optimization problem.

Stopping criteria for the ADMM iterations can be constructed from convergence of the coupled variables to common values [21]. However, in the hybrid case, where optimality and convergence to agreement on coupled variables is not guaranteed, we will use finite termination with a range of fixed numbers of iterations, exploring the effect of this finite termination on performance. The reader is referred to [21] for a detailed overview of ADMM, to [8] for an example of distributed MPC via ADMM for linear systems, and to [2] for an example of ADMM being applied to distributed hybrid MPC.

Note that in the ADMM algorithm there is a tuning parameter $\rho > 0$. For all experiments in this paper this value was set to $\rho = 0.5$, as this was found to perform the best.

V. EXPERIMENTS

In this section we compare the five controllers presented in Section IV on three instances the benchmark problem. The first and simplest instance, referred to as task 1, considers the front vehicle as the leader, a constant speed reference trajectory, homogeneous vehicles, and a constant distance spacing policy. With task 1 we explore the performance and complexity of the controllers under the different prediction models (Model I and II) and norm choices $\|\cdot\|$ in the costs. The second instance, task 2, considers an aggressive variable-speed reference trajectory, inhomogeneous vehicles, a velocity-dependent spacing policy, and again the front vehicle as leader. Finally, the third instance, task 3, is identical

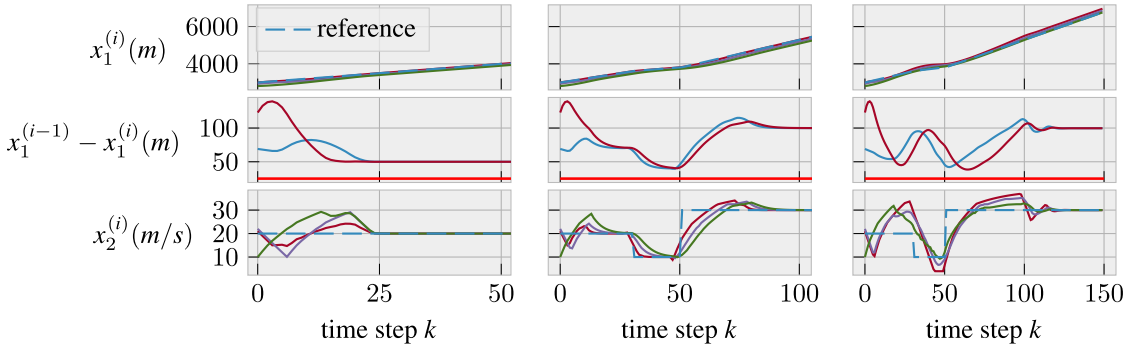
[2]We also implemented and tested simple vehicle speed prediction algorithms using linear function and saturation function predictors. We found that, in our simulations, the constant speed assumption outperformed both these. As the main focus of this work is the formulation and proposal of the benchmark problem, an exploration of vehicle speed prediction algorithms is left to future work, and only constant speed predictions are used for the decentralized controller.

TABLE 2. Experimental parameters.

Parameter	g	μ	c	v_{MIN}	v_{MAX}	a_{MIN}	a_{MAX}	u_{MAX}	d_{safe}	p_{MIN}	p_{MAX}	Q_x	Q_u	T
Value	9.8	0.01	0.5	3.94	45.84	-2	2.5	1	25	0	10000	$\begin{bmatrix}1 & 0\\ 0 & 0.1\end{bmatrix}$	1	1

Parameter	task 1 (d_0,t_0,l)	task 2 (d_0,t_0,l)	task 3 (d_0,t_0,l)	task 1 m_i	task 2 m_i	task 3 m_i
Value	$(50, \text{NA}, 1)$	$(10, 3, 1)$	$(10, 3, \mathcal{M} \setminus \{1\})$	$m_i = 800$	$m_i \sim \mathcal{U}(700, 1000)$	$m_i \sim \mathcal{U}(700, 1000)$

Parameter	Initial positions	Initial velocities
Value	$x_1^{(i-1)}(0) - x_1^{(i)}(0) \sim \mathcal{U}(60, 160)$	$x_2^{(i)}(0) \sim \mathcal{U}(5, 35)$



(a) task 1 (b) task 2 (c) task 3, $l = 3$

FIGURE 4. Platoon formation under the centralized controller for the 3 tasks with $M = 3$ and $N = 6$., Position (top), inter-vehicle spacing (middle) with safe distance (red line), and velocity (bottom).

to task 2, except that the controllers are evaluated with each vehicle as the leader, excluding the front vehicle. On tasks 2 and 3 we explore the performance and characteristics of the controllers as the size of the platoon (M) and the length of the prediction horizon (N) vary. All tasks consider each vehicle to be initialized with a randomized velocity, distributed uniformly in the range $\begin{bmatrix}5 & 35\end{bmatrix}$ ms^{-1}, and randomized positions behind the preceding vehicle, distributed uniformly in the range $\begin{bmatrix}60 & 160\end{bmatrix}$ m.

Table 2 gives all numerical coefficients that are used in the experiments for the tasks, controllers, and constraints. Figure 4 demonstrates representative platoon trajectories for the three tasks. We highlight that these are just three of many possible configurations for the benchmark problem, with the tuning knobs, outlined in Section III, providing a way to construct many more instances of varying difficulty.

We consider the following performance indicators:

- Tracking performance J (24).
- Tracking performance with respect to the centralized controller $\Delta J = J - J_{\text{cent}}$. As the distributed approaches do not provide formal bounds on suboptimality, this indicator is used to empirically explore the performance loss introduced by a distributed controller, with respect to the performance of the centralized MPC controller.
- Computation time $t_{\text{COMP}} = (t_{\min}, t_{\text{av}}, t_{\max})$. This triple contains the minimum, average, and maximum time required to compute the control inputs for a simulation step. These times are for the *entire platoon*, i.e., for the decentralized controller, the computation time for a time step is the maximum computation time required to compute a local MPC solution, as the subsystems solve their subproblems in parallel, while for the sequential controller, it is the sum of the local computation times in the sequential order. Note that, as all distributed controllers rely on solving mixed-integers programs, the worst-case computational complexity of each controller is exponential in practice [9]. In practice, however, the different handling of coupling and communication between algorithms can result in significant differences in the observed computation times. This is explored thoroughly in the following.
- Number of explored nodes n_{no}. This indicator serves as a proxy for the local memory required by each controller. It is determined by the maximum number of nodes that are explored in one BnB procedure.

In all simulations, the continuous-time nonlinear hybrid model (1) is used to simulate the underlying system. All mixed-integer programs are solved using Gurobi [10]. All simulations are run on a Linux machine using five AMD EPYC 7252 cores, 1.38GHz clock speed, and 251Gb of RAM. We note that this hardware is significantly more powerful than that which is typically available onboard a vehicle. As the scope of this work is to present a benchmark control problem for the comparison of control strategies, the use of overpowered hardware does not detract from the relevance of the comparative results. Naturally, for an onboard solution, considerations such as embedded feasibility and electronic control unit constraints need to be taken into account. In the following, for the iterative controllers, the number of iterations is presented along with the controller name, e.g., ADMM(20) represents the ADMM-based controller with 20 iterations.

A. LINEAR VERSUS QUADRATIC COSTS

First, we compare the use of 1- and 2-norm costs, $\|\cdot\| = \|\cdot\|_1$ and $\|\cdot\| = \|\cdot\|_2^2$, respectively. Figure 5 compares the platoon trajectory on task 1, with a centralized controller, for each norm. Quadratic, 2-norm-based, costs penalize large tracking errors heavily, and small tracking errors lightly, while 1-norm-based costs penalize the tracking errors linearly. As a result, the platoon formation differs significantly between the two. A 2-norm-based cost encourages the fleet to cooperate and form the platoon earlier, while moving towards the reference trajectory.

Figure 6 shows a heat-map comparison of the average computation time t_{av} for each controller under each norm, as the prediction horizon and size of the network are increased. The heat map shows that for the sequential and decentralized controllers, the 1-norm-based cost provides a computational benefit, while for the other controllers, it is the 2-norm-based cost that provides the speed up. It is often cited that, when solving mixed-integer programs, 1-norm-based costs are preferred, as MILPs can be solved faster than MIQPs by commercial BnB solvers [25]. We highlight, however, that the time required to solve a mixed-integer program varies significantly from problem to problem, based on the initial conditions, warm-start solution, and the effectiveness of the online branching and pruning procedures. As the different norms lead to different trajectories, and therefore different mixed-integer optimization problems, the choice should be explored explicitly for different controllers and control tasks. In the remaining experiments in this paper, we use costs based on the 2-norm, as the faster convergence to a regularly spaced platoon is preferred.

B. MODEL COMPARISON

We compare the performance and computational complexity of the prediction models; Model I and Model II, using task 1. Additionally, we compare the hybrid models against the use of a discretization of the nonlinear hybrid dynamics (1) as a prediction model.[3]

Table 3 gives the percentage difference between the models' tracking performances and average computation times on task 1 with $M = 3$ and $N = 5$. For all controllers, the tracking performance is improved by using Model II over Model I. This is in line with the observation that Model II, by modeling the gear choice as a discrete input, retains the use of a larger subset of the velocity-gear decision space. As Model II is not restricted to have a one-to-one mapping between velocity and gear, for a given velocity there may be a gear choice available to Model II that is not available to Model I, which may improve tracking by, e.g., giving a faster acceleration. Hence, for a given state, any optimal solution to the MPC optimization problem using Model I provides an upper bound on the cost of the optimal solution when using Model II.

The average computation time required for Model II is significantly higher than that for Model I. This is particularly severe for the control strategies that use MPC controllers with decision spaces covering more than one vehicle, i.e., the centralized and event-based controllers. This large increase in computational complexity indicates that Model I, the PWA model, contains some structure that is advantageous in the BnB process. Indeed, for the centralized controller, the optimal control problem at each time step contains 105 and 120 binary decision variables for Model I and Model II, respectively. However, following the Gurobi pre-solve procedure,[4] on average the numbers of remaining binary decision variables are 20 and 52 for Model I and Model II, respectively. While slightly decreasing performance via an artificial restriction of the velocity-gear decision space,Model I offers significant computational advantages through its PWA structure. An additional advantage of Model I is that it can be converted to many equivalent modeling frameworks [35], for which alternative control strategies could be explored. These alternative strategies are beyond the scope of this paper and are left to future work. In contrast, using Model II as a prediction model will always result in a mixed-integer program, as the gears are explicitly modeled as discrete inputs. In the following experiments, we use Model I for all prediction models due to the significantly lower computational burden.

The nonlinear prediction model gives a slight improvement in performance over Model II, as it captures the true quadratic friction. However, the addition of the non-convex constraints in the dynamics results in a vastly increased complexity. Again this is particularly extreme for the centralized and event-based controllers, which solve larger optimization problems. This motivates the use of a hybrid approximation for the nonlinear friction, sacrificing very little performance for a huge computational speed-up.

C. PERFORMANCE COMPARISON - TASK 2

We compare the performance of the controllers on task 2 as the number of vehicles M varies, exploring how the complexity of each controller scales as the size of the platoon increases. Furthermore, we compare their performance as the prediction horizon N varies. For the platoon size, task 2 is simulated for a fixed prediction horizon of $N = 6$, with the number of vehicles varying from $M = 2$ to $M = 10$. Figure 7 presents the performance indicators over ten randomized initial conditions and vehicle masses (see Table 2). The relative tracking performance ΔJ shows that, in general, the non-centralized controllers introduce a performance drop with respect to centralized control and that, as the number of vehicles increases, the extent of

[3]Gurobi supports quadratic equality constraints and is thus able to solve the mixed-integer nonlinear programs (MINLP) that arise when using (1) as a prediction model. We also compared Gurobi against the dedicated MINLP solver MindtPy [45] for solving the control problems, and found Gurobi to be faster.

[4]The Gurobi pre-solve procedure attempts to simplify an optimization problem, prior to solving it. This process involves many complex operations, such as the removal of redundant constraints and variables, detection of implied bounds on variables, cut generation, etc.

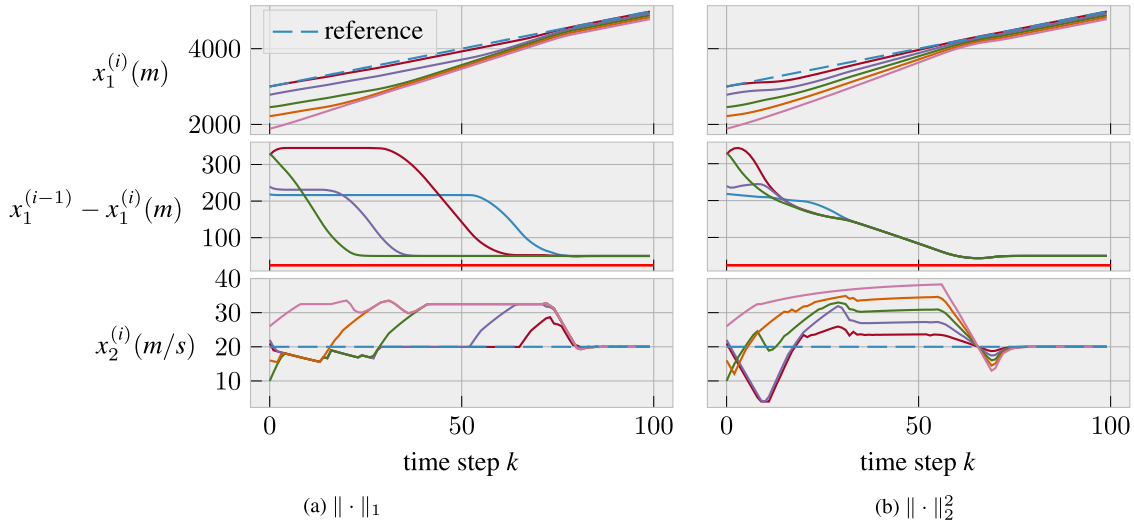


FIGURE 5. Platoon formation under the centralized controller with 1- and 2-norm costs for $M = 5$ and $N = 6$. Position (top), inter-vehicle spacing (middle) with safe distance (red line), and velocity (bottom).

TABLE 3. Comparison between prediction models for task 1 with $M = 3$, $N = 5$.

	Centralized	Decentralized	Sequential	Event (4)	ADMM (20)
$\frac{J_{\text{II}}-J_{\text{I}}}{J_{\text{I}}} \times 100$	−4.06	−3.77	−4.03	−4.05	−3.64
$\frac{t_{\text{av, II}}-t_{\text{av, I}}}{t_{\text{I}}} \times 100$	1.58e4	243.20	247.28	1.42e4	159.42
$\frac{J_{\text{NL}}-J_{\text{I}}}{J_{\text{I}}} \times 100$	−4.56	−4.14	−4.54	−4.26	−4.06
$\frac{t_{\text{av, NL}}-t_{\text{av, I}}}{t_{\text{I}}} \times 100$	5.32e5	1.36e3	1.63e3	6.96e4	813.00

this performance drop worsens. The event-based controllers perform the best for small numbers of vehicles, achieving centralized performance. As the number of vehicles increases a performance drop is seen. However, this performance drop is not uniformly increasing with the number of vehicles, and is not uniformly improved by the number of iterations. Particularly noticeable, the event(4) controller performs the best on average for $N = 7$ to $N = 9$, and the worst for $N = 10$. This demonstrates the heuristic nature of this approach, as the performance drop is inconsistent, and more iterations of the algorithm do not necessarily result in a better performance. In contrast, the other non-centralized controllers introduce a performance drop for all platoon sizes, that worsens as the number of vehicles increases. The decentralized controller introduces extreme performance drops for large numbers of vehicles, demonstrating the importance of communication.

The computation times t_{COMP} show that in general, as the number of vehicles increases, the computational complexity of the controllers increases. Although the non-centralized controllers solve subproblems whose number of decision variables is independent of the platoon size, the computational requirements increase with the size of the platoon. This demonstrates how the complexity of solving mixed-integer programs can vary with the complexity of the control challenge. Only the decentralized, sequential, and ADMM(5) controllers maintained a maximum computation time for the fleet $t_{\max}$ of less than the sampling time $T = 1$s for all platoon sizes. The event-based controllers in particular show extreme computational complexities, reaching $t_{\max}$ greater than 100s. Note that the computation times for the event-based controllers are similar across numbers of iterations. This shows that the extra iterations are used infrequently, as the extra iterations are used only in the event of a cost improvement (see Section IV).

The node count n_{no} shows that the local memory required by the controllers is independent of the platoon size for the decentralized, sequential, and ADMM controllers. For the event-based controllers, the amount of local memory required is significant, with respect to the other controllers, as the local mixed-integer problems consider the decision variables of adjacent vehicles, and the memory requirements scale poorly with the number of vehicles.

For the varying prediction horizon, task 2 is simulated for a fixed platoon size of $M = 6$, with the prediction horizon varying from $N = 2$ to $N = 6$. Figure 8 presents the performance indicators over ten randomized initial conditions and vehicle masses (see Table 2). The tracking performance shows that as the size of the prediction horizon increases, in general, the performance of the non-centralized controllers improves. A clear exception is the decentralized controller, which performs worse as the horizon increases. Exploring this further, Figure 9 shows the decentralized performance as N increases for $M = 2$. The opposite trend is observed,

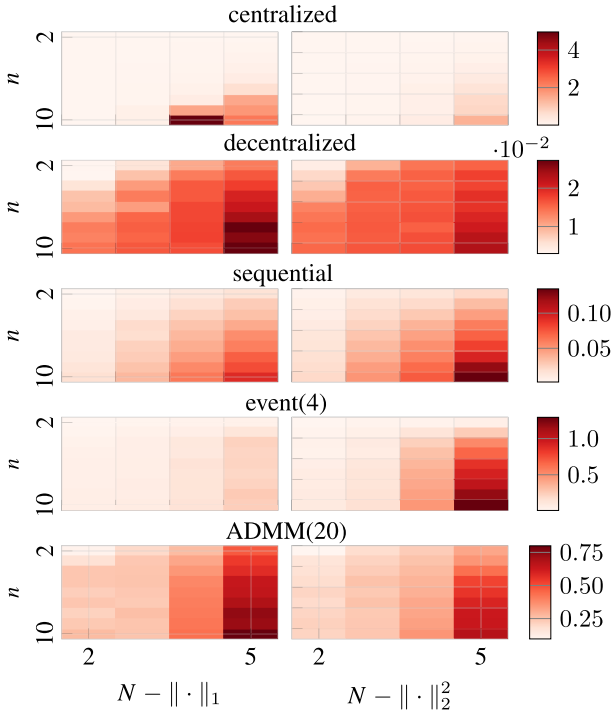


FIGURE 6. Heat map for the average computation time of each controller, in seconds, under 1- and 2-norm stage costs. The length of the prediction horizon increases along the x-axis, while the size of the network increases down the y-axis.

indicating a non-trivial relationship between the horizon length and platoon size for the decentralized controller. This may result from the decentralized controller relying heavily on extrapolated trajectory estimates, such that for some platoon sizes, a shorter prediction horizon becomes beneficial, reducing prediction uncertainty. The computation times t_{COMP} show exponential increases in computational complexity as the prediction horizon increases (linear trend on a log scale). This demonstrates how the worst-case complexity of a mixed-integer program grows exponentially with the number of integer decision variables [9], and that as the prediction horizon increases, the subproblems have increasing numbers of binary decision variables. Similarly, the node counts n_{no} shows that the local memory requirements also grow exponentially as the prediction horizon increases. These effects are most significant in the event-based controllers, where the subproblems consider the decision variables associated with more than one vehicle. The computation time and memory requirements for the event-based controllers are nearly identical, indicating that extra iterations were rarely used.

All controllers satisfied the safety distance constraints for all task 2 simulations.

Note that the above results are additionally available in tabular form in Appendix A.

D. PERFORMANCE COMPARISON - TASK 3

We compare the performance of a subset of the controllers on task 3 as the size of the platoon M increases. Task 3 is

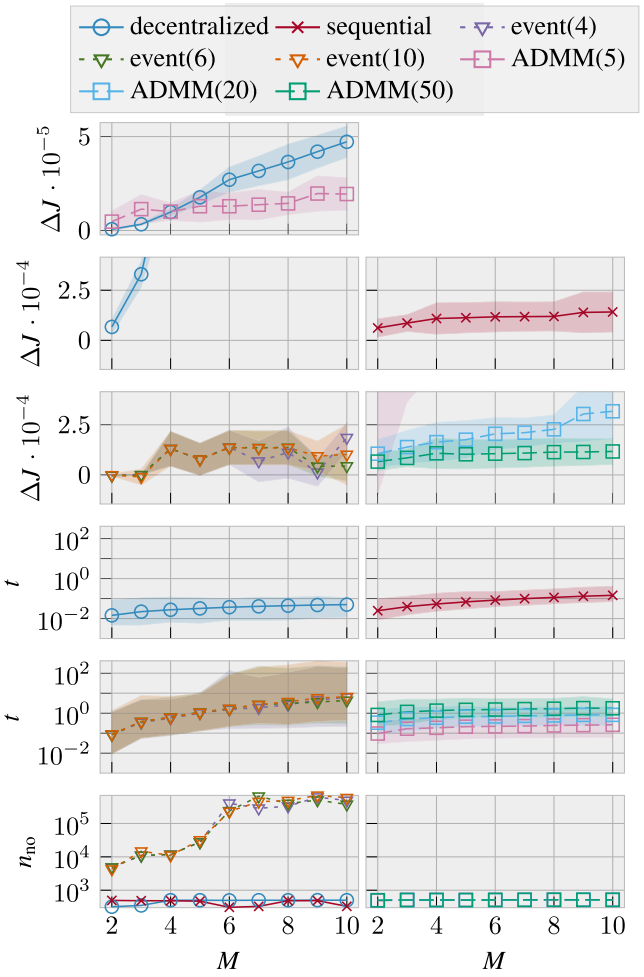


FIGURE 7. Task 2 with prediction horizon $N = 6$. Rows 1-3: average relative closed-loop tracking performance ΔJ (a standard deviation in shaded region). Rows 2 and 3 share a y-axis scaling such that controllers in different plots can be compared. As the decentralized and ADMM(5) controllers have vastly increased performance drops, Row 1 shows a zoomed out view for these. Rows 4-5: average computation time t_{av} (t_{max} and t_{min} in shaded region) - log scale. Row 6: node count n_{no} - log scale.

simulated with a prediction horizon of $N = 6$ and the number of vehicles varying from $M = 2$ to $M = 5$. For each value of M, a simulation is run where each vehicle, except the front vehicle, is the leader, i.e., $l \in \mathcal{M} \setminus \{1\}$. For task 3, where all vehicles can be the leader, we note that the sequential controller is often unable to track the reference trajectory, due to the strict agreement enforced on coupled states. In the sequential approach, solutions for the coupled variables are passed along the sequence of vehicles, with no opportunity given for negotiation. As such, with the leader 'sandwiched' in the middle of the platoon, it may be unable to coerce the platoon towards the reference trajectory, as the other vehicles, focusing on tracking adjacent vehicles, may restrict the leader. Figure 10 demonstrates an example of this. While the leader $l = 4$ tries to speed up to reach the reference trajectory, pushing up against its preceding vehicle, the vehicles at the front of the platoon slow down to track each other. By the velocity-dependent spacing policy, the desired

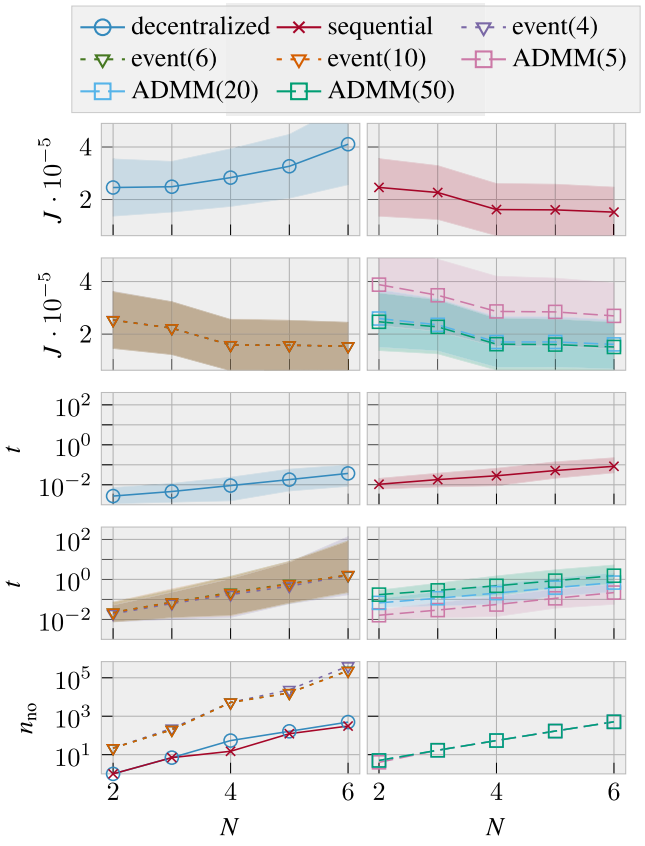


FIGURE 8. Task 2 with number of vehicles $M = 6$. Rows 1-2: average closed-loop tracking performance J (a standard deviation in shaded region). Rows 3-4: average computation time t_{av} (t_{max} and t_{min} in shaded region) - log scale. Row 5: node count n_{no} - log scale.

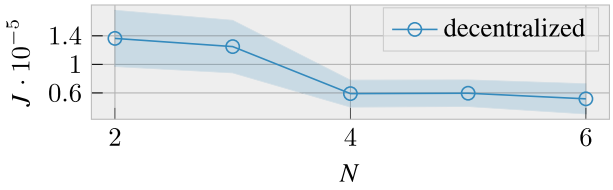


FIGURE 9. Average closed-loop tracking performance J (a standard deviation in shaded region) for the decentralized controller on task 2 with $M = 2$.

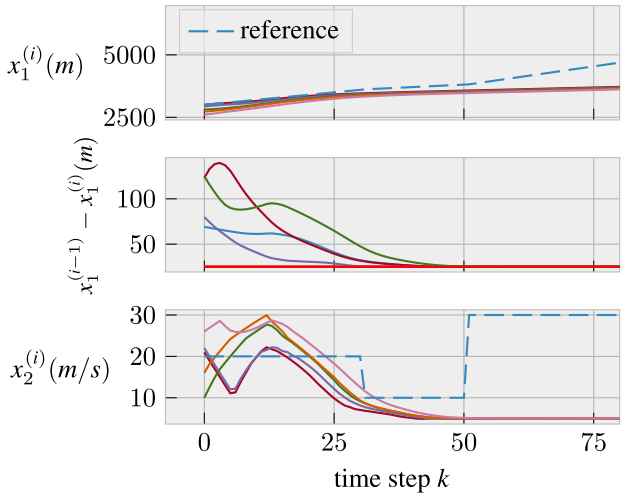


FIGURE 10. Sequential controller for task 3 with $M = 5$, $N = 6$, and $l = 4$. Position (top), inter-vehicle spacing (middle) with safe distance (red line), and velocity (bottom).

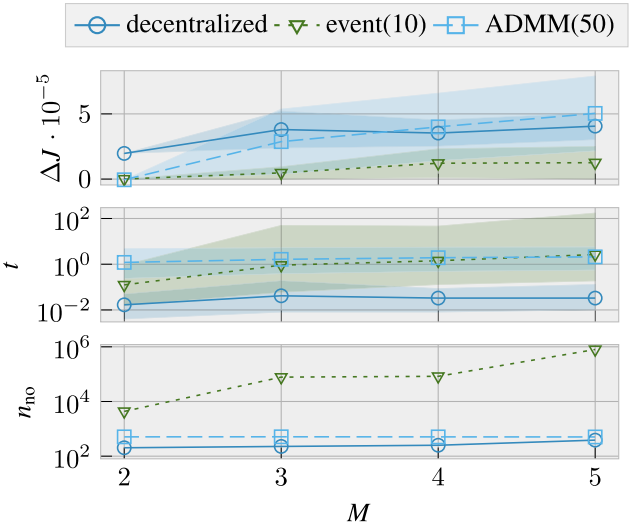


FIGURE 11. Task 3 with prediction horizon $N = 6$. Row 1: average relative closed-loop tracking performance J (a standard deviation in shaded region). Row 2: average computation time t_{av} (t_{max} and t_{min} in shaded region) - log scale. Row 3: node count n_{no} - log scale.

spacing shrinks as the vehicles slow down; causing them to slow down more and trapping the leader, up to the point where a deadlock is reached and the leader cannot initiate any acceleration, as it is at the safe distance limit with its adjacent vehicles.

Figure 11 presents the performance indicators averaged over all leader choices. The relative tracking performance shows that as the number of vehicles increases the performance drop of the non-centralized controllers increases. Furthermore, the performance drop is more severe than for task 2, demonstrating the added difficulty of task 3. In particular, on this more complex task, the ADMM(50) controller is outperformed by the decentralized controller for $M \geq 4$, demonstrating how ADMM in the non-convex setting gives no guarantee of approaching the centralized optimum, with the suboptimality worsening with task complexity. In Appendix B this is discussed further by looking at convergence of agreement on coupled variables within a time step. The event-based controller performs the best; however, the performance drop scales poorly with the number of vehicles, and the complexity of the controller can be seen to explode even for modest numbers of vehicles, with $t_{max} > 100$ even for $M = 5$. The inter-vehicle safe distance was maintained successfully for all task 3 simulations.

E. COMPARISON SUMMARY

In this section we provide a summary of the controller comparison, focusing on the suboptimality and computational

complexity of each controller, in particular as the size of the platoon increases, giving general guidelines of when each controller is preferred. The decentralized controller presents the best option from the perspective of complexity, consistently requiring the least computation time and memory, thanks to subproblems being solved just once in parallel. Furthermore, the decentralized controller requires no communication, thus removing the overhead of inter-vehicle communication infrastructure. The cost for this low complexity is lower tracking performance, with the distributed controller introducing a performance drop that scales poorly with the size of the platoon. However, when computational power is limited, or communication is unavailable, the decentralized controller is preferred.

The sequential controller presents a slightly higher complexity than the decentralized controller, as the subproblems are solved in series rather than in parallel. Showcasing the benefit of even limited communication, the performance drop of the sequential controller is far smaller than that of the decentralized controller, particularly for large platoons. However, the sequential controller's rigid agreement mechanism renders it inapplicable when the leader is not the front vehicle. Therefore, when communication is available and the leader is the front vehicle, the sequential controller is recommended, giving a modest performance drop with a modest computational burden.

The event-based controllers overall give the best tracking performance. However, the computational complexity is severe, with extremely large computation times and memory requirements, scaling poorly with the size of the platoon. Furthermore, the performance does not necessarily improve with increasing numbers of iterations, potentially requiring manual tuning as platoon sizes and tasks vary. Additionally, the event-based controllers optimize over the trajectories of adjacent vehicles, requiring full knowledge of their dynamics. Finally, communicating the updated solutions between iterations requires communication between vehicles up to two hops away. Consequently, the event-based controllers are recommended when good tracking performance is desired at the expense of restrictive requirements on communication infrastructure, complexity, and information sharing.

The ADMM controllers present a complexity that increases with the number of algorithm iterations, with high numbers of iterations needed to achieve modest drops in tracking performance. The iterative mechanism allows for negotiation, and the ADMM controllers can be applied when the leader is not the front vehicle. As such, the ADMM controller is preferred over the sequential controller only when the leader is not the front vehicle.

VI. CONCLUSION AND CONCLUDING REMARKS

In this work we have presented a case study to serve as a benchmark problem for comparing distributed MPC controllers for hybrid systems. We have considered control of a platoon of vehicles with explicit gear management. Two hybrid vehicle modeling options have been presented, the benchmark problem has been specified, and five existing hybrid MPC approaches have been evaluated on the benchmark.

The performance of the evaluated existing controllers highlights the need for new distributed hybrid MPC strategies that can approach centralized performance, even as the size of the system increases. In particular, distributed controllers that strike a favorable trade-off between computational intensity and performance are clearly lacking. Indeed, one point of note is that all controllers evaluated require the online solution to a mixed-integer linear or quadratic optimization problem. This limits the applicability of these approaches as the scale of the problem increases. We highlight that there is huge potential for the development of distributed hybrid MPC controllers that avoid solving mixed-integer programs online altogether.

The benchmark problem presented in this work can be modified through the tuning knobs: platoon size, reference trajectory, spacing policy, homogeneity, and leader position, for a modular and adaptable control problem, upon which future distributed hybrid MPC controllers can be evaluated. Future work will, in addition to developing controllers that fill the gaps outlined above, involve exploring distributed hybrid MPC strategies for this problem that can provide theoretical guarantees on recursive feasibility and stability, as well as string stability of the network. Furthermore, future work will look to extend the benchmark to consider non-ideal communication, such as quantization, communication delays, and asynchronous updates, as well as mixed-traffic scenarios. Finally, future work will also consider real-world versions of this benchmark in field experiment test beds.

APPENDIX A
TABULAR DATA

Tables 4 and 5 provide the results of Figures 7 and 8 in tabular form.

APPENDIX B
ADMM CONVERGENCE

To elaborate on how non-convergence of ADMM in the non-convex setting can worsen the performance of the distributed ADMM controller, Figure 12 shows the primal residual, a norm measurement of the disagreement between coupled states, and the dual residual, a measure of optimality in the dual objective [21], across ADMM iterations for a representative time step within task 3. We use 500 ADMM iterations to demonstrate a lack of convergence. In particular, it can be seen that the sharp drop in performance of the ADMM(50) controller in task 3 as the number of vehicles increases is due to larger primal residual values, i.e., more disagreement on the coupled states.

TABLE 4. Data from Figure 7. Cost ΔJ, computation time t (seconds), and node count n_{no} as a function of the horizon length M for each controller. Values reported as mean ± standard deviation.

Controller	$M=2$	$M=3$	$M=4$	$M=5$	$M=6$	$M=7$	$M=8$	$M=9$	$M=10$
					Cost $\Delta J \times 10^{-4}$				
decentralized (○)	0.67 ± 0.38	3.30 ± 0.66	9.79 ± 2.24	17.7 ± 4.02	27.0 ± 6.72	31.7 ± 7.91	36.5 ± 9.45	42.0 ± 8.72	47.2 ± 8.40
sequential (×)	0.619 ± 0.456	0.862 ± 0.420	1.09 ± 0.783	1.13 ± 0.755	1.17 ± 0.751	1.18 ± 0.730	1.19 ± 0.729	1.40 ± 1.02	1.42 ± 0.999
event(4) (▽)	-0.029 ± 0.074	≈ 0	1.30 ± 0.864	0.781 ± 0.802	1.38 ± 0.857	0.685 ± 1.03	1.12 ± 1.28	0.127 ± 0.697	1.86 ± 0.750
event(6) (▽)	-0.029 ± 0.074	≈ 0	1.30 ± 0.864	0.780 ± 0.802	1.36 ± 0.851	1.36 ± 0.846	1.37 ± 0.854	0.423 ± 0.492	0.443 ± 0.756
event(10) (▽)	-0.004 ± 0.066	-0.103 ± 0.392	1.30 ± 0.863	0.778 ± 0.803	1.36 ± 0.848	1.34 ± 0.851	1.35 ± 0.855	0.939 ± 0.749	1.02 ± 1.51
ADMM(5) (□)	4.71 ± 5.68	11.4 ± 7.74	10.0 ± 4.69	12.9 ± 8.44	12.9 ± 7.08	13.7 ± 8.06	14.4 ± 6.01	19.7 ± 9.39	19.4 ± 8.64
ADMM(20) (□)	1.06 ± 0.723	1.39 ± 0.827	1.64 ± 0.987	1.76 ± 0.952	2.05 ± 0.808	2.11 ± 0.732	2.27 ± 0.716	3.04 ± 1.39	3.18 ± 1.38
ADMM(50) (□)	0.664 ± 0.448	0.850 ± 0.444	1.08 ± 0.770	1.03 ± 0.732	1.06 ± 0.718	1.09 ± 0.710	1.14 ± 0.702	1.14 ± 0.673	1.17 ± 0.652
					Computation time t [s]				
decentralized (○)	0.014 ± 0.040	0.022 ± 0.052	0.028 ± 0.056	0.032 ± 0.052	0.037 ± 0.045	0.041 ± 0.055	0.044 ± 0.054	0.048 ± 0.060	0.050 ± 0.049
sequential (×)	0.025 ± 0.045	0.039 ± 0.058	0.053 ± 0.081	0.068 ± 0.097	0.084 ± 0.094	0.100 ± 0.109	0.115 ± 0.126	0.131 ± 0.155	0.145 ± 0.170
event(4) (▽)	0.085 ± 0.594	0.344 ± 2.30	0.558 ± 2.47	1.00 ± 4.88	1.56 ± 68.9	1.84 ± 31.2	2.65 ± 55.5	4.57 ± 113	6.23 ± 94.1
event(6) (▽)	0.084 ± 0.547	0.377 ± 2.40	0.632 ± 3.18	1.12 ± 5.14	1.65 ± 44.0	2.67 ± 111	3.06 ± 89.9	3.60 ± 120	4.31 ± 106
event(10) (▽)	0.089 ± 0.659	0.374 ± 3.75	0.611 ± 3.46	1.08 ± 4.57	1.70 ± 41.2	2.61 ± 101	3.71 ± 129	5.69 ± 207	6.56 ± 175
ADMM(5) (□)	0.095 ± 0.188	0.167 ± 0.247	0.185 ± 0.245	0.211 ± 0.268	0.218 ± 0.264	0.226 ± 0.262	0.243 ± 0.267	0.252 ± 0.259	0.262 ± 0.266
ADMM(20) (□)	0.335 ± 0.779	0.511 ± 1.02	0.595 ± 1.04	0.671 ± 1.09	0.691 ± 0.939	0.733 ± 1.01	0.769 ± 0.979	0.822 ± 1.02	0.825 ± 1.02
ADMM(50) (□)	0.816 ± 1.78	1.16 ± 2.40	1.34 ± 2.31	1.47 ± 2.56	1.53 ± 2.35	1.60 ± 2.41	1.68 ± 2.47	1.79 ± 3.05	1.75 ± 2.29
					Node count n_{no}				
decentralized (○)	320	350	501	501	496	501	500	503	502
sequential (×)	495	484	483	469	308	328	484	487	329
event(4) (▽)	4877	10996	11786	27701	403789	284882	328714	614077	487958
event(6) (▽)	4877	10996	11786	27701	237830	651889	395747	499950	373212
event(10) (▽)	4273	14713	11099	31890	227906	450838	482663	694637	590075
ADMM(5) (□)	484	515	513	513	513	514	515	515	517
ADMM(20) (□)	503	510	509	507	509	515	517	518	515
ADMM(50) (□)	499	509	509	507	512	515	517	515	515

TABLE 5. Data from Figure 8. Cost J, computation time t (seconds), and node count n_{no} as a function of the depth N for each controller. Values reported as mean ± standard deviation.

Controller	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$
			Cost $J \times 10^{-5}$		
decentralized (○)	2.46 ± 1.09	2.49 ± 0.967	2.83 ± 1.10	3.27 ± 1.22	4.11 ± 1.54
sequential (×)	2.46 ± 1.10	2.27 ± 1.03	1.62 ± 0.995	1.61 ± 0.980	1.52 ± 0.955
event(4) (▽)	2.54 ± 1.09	2.23 ± 1.01	1.58 ± 0.980	1.57 ± 0.952	1.54 ± 0.914
event(6) (▽)	2.53 ± 1.09	2.23 ± 1.01	1.58 ± 0.980	1.58 ± 0.953	1.54 ± 0.914
event(10) (▽)	2.53 ± 1.09	2.23 ± 1.01	1.58 ± 0.980	1.58 ± 0.953	1.54 ± 0.914
ADMM(5) (□)	3.88 ± 1.56	3.48 ± 1.38	2.86 ± 1.34	2.84 ± 1.29	2.69 ± 1.26
ADMM(20) (□)	2.59 ± 1.10	2.35 ± 0.996	1.70 ± 0.968	1.70 ± 0.962	1.61 ± 0.939
ADMM(50) (□)	2.47 ± 1.10	2.28 ± 1.03	1.62 ± 0.988	1.60 ± 0.973	1.51 ± 0.951
			Computation time t [s]		
decentralized (○)	0.0027 ± 0.0031	0.0046 ± 0.0052	0.0091 ± 0.0113	0.0182 ± 0.0281	0.0371 ± 0.0452
sequential (×)	0.0106 ± 0.0075	0.0182 ± 0.0152	0.0282 ± 0.0292	0.0517 ± 0.0604	0.0841 ± 0.0939
event(4) (▽)	0.0188 ± 0.0220	0.0618 ± 0.1091	0.172 ± 0.537	0.456 ± 3.571	1.56 ± 68.9
event(6) (▽)	0.0222 ± 0.0325	0.0726 ± 0.1455	0.221 ± 0.725	0.611 ± 3.971	1.65 ± 44.0
event(10) (▽)	0.0221 ± 0.0332	0.0712 ± 0.1619	0.205 ± 0.692	0.577 ± 3.973	1.70 ± 41.2
ADMM(5) (□)	0.0160 ± 0.0104	0.0291 ± 0.0282	0.0550 ± 0.0649	0.112 ± 0.144	0.218 ± 0.264
ADMM(20) (□)	0.0667 ± 0.0361	0.114 ± 0.099	0.206 ± 0.277	0.387 ± 0.535	0.691 ± 0.939
ADMM(50) (□)	0.172 ± 0.096	0.282 ± 0.262	0.476 ± 0.675	0.864 ± 1.331	1.53 ± 2.35
			Node count n_{no}		
decentralized (○)	1	7	54	164	496
sequential (×)	1	7	15	123	308
event(4) (▽)	22	232	5206	23692	403789
event(6) (▽)	22	197	5206	16286	237830
event(10) (▽)	22	197	5206	16286	227906
ADMM(5) (□)	4	17	54	169	513
ADMM(20) (□)	5	17	55	169	509
ADMM(50) (□)	5	17	54	169	512

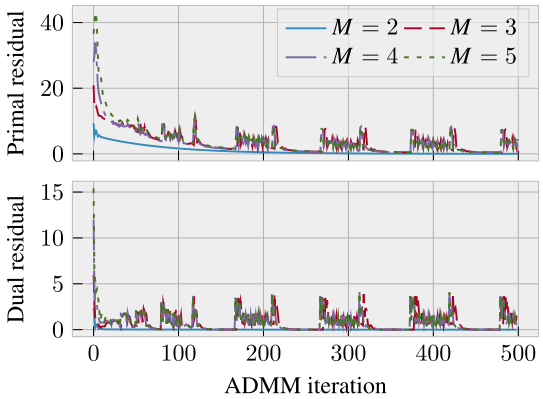


FIGURE 12. Primal and dual residuals during ADMM iterations for task 3.

The lack of guarantees for ADMM in the non-convex case can also be seen, as the residuals do not converge to zero even when allowing an extreme number of ADMM iterations.

REFERENCES

[1] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annu. Rev. Control*, vol. 43, pp. 1–64, Apr. 2017.

[2] X. Luan, B. De Schutter, L. Meng, and F. Corman, "Decomposition and distributed optimization of real-time traffic management for large-scale railway networks," *Transp. Res. B, Methodol.*, vol. 141, pp. 72–97, Nov. 2020.

[3] P. R. C. Mendes, J. M. Maestre, C. Bordons, and J. E. Normey-Rico, "A practical approach for hybrid distributed MPC," *J. Process Control*, vol. 55, pp. 30–41, Jul. 2017.

[4] H. van Ekeren, R. R. Negenborn, P. J. van Overloop, and B. De Schutter, "Time-instant optimization for hybrid model predictive control of the Rhine–Meuse delta," *J. Hydroinformatics*, vol. 15, no. 2, pp. 271–292, 2013.

[5] J. M. Maestre, L. Raso, P. J. van Overloop, and B. De Schutter, "Distributed tree-based model predictive control on a drainage water system," *J. Hydroinformatics*, vol. 15, no. 2, pp. 335–347, Apr. 2013.

[6] Z. Su, A. Jamshidi, A. Núñez, S. Baldi, and B. D. Schutter, "Distributed chance-constrained model predictive control for condition-based maintenance planning for railway infrastructures," in *Predictive Maintenance in Dynamic Systems*. Cham, Switzerland: Springer, 2019, pp. 533–554.

[7] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[8] T. H. Summers and J. Lygeros, "Distributed model predictive consensus via the alternating direction method of multipliers," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2012, pp. 79–84.

[9] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

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

[11] *V12. 1: User's Manual for CPLEX*, IIBM ILOG, New York, Jan. 2009.

[12] D. Groß and O. Stursberg, "Distributed predictive control for a class of hybrid systems with event-based communication," *IFAC Proc. Volumes*, vol. 46, no. 27, pp. 383–388, 2013.

[13] A. Richards and J. P. How, "Robust distributed model predictive control," *Int. J. Control*, vol. 80, no. 9, pp. 1517–1531, Sep. 2007.

[14] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Distributed robust receding horizon control for multivehicle guidance," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 4, pp. 627–641, Jul. 2007.

[15] R. B. Larsen, B. Atasoy, and R. R. Negenborn, "Model predictive control with memory-based discrete search for switched linear systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6769–6774, 2020.

[16] A. Ma, D. Li, and Y. Xi, "Distributed MPC based on robustly controllable sets for PWA systems," *Automatica*, vol. 154, Aug. 2023, Art. no. 111078.

[17] A. Ma, D. Li, and Y. Xi, "Robust tube-based distributed MPC for PWA systems," *IET Control Theory & Appl.*, vol. 17, no. 16, pp. 2205–2214, 2023.

[18] R. Vujanic, P. Mohajerin Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari, "A decomposition method for large scale MILPs, with performance guarantees and a power system application," *Automatica*, vol. 67, pp. 144–156, May 2016.

[19] A. Falsone, K. Margellos, and M. Prandini, "A decentralized approach to multi-agent MILPs: finite-time feasibility and performance guarantees," *Automatica*, vol. 103, pp. 141–150, May 2019.

[20] A. Camisa, I. Notarnicola, and G. Notarstefano, "Distributed primal decomposition for large-scale MILPs," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 413–420, Jan. 2022.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jul. 2011.

[22] Z. Liu and O. Stursberg, "Distributed solution of mixed-integer programs by ADMM with closed duality gap," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 279–286.

[23] C. Beltran and F. J. Heredia, "Unit commitment by augmented Lagrangian relaxation: Testing two decomposition approaches," *J. Optim. Theory Appl.*, vol. 112, no. 2, pp. 295–314, Feb. 2002.

[24] *SMART Website*. Accessed: Dec. 11, 2023. [Online]. Available: https://uk.smart.com/en/

[25] D. Corona and B. De Schutter, "Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 2, pp. 365–372, Mar. 2008.

[26] S. E. Li, Y. Zheng, K. Li, Y. Wu, J. K. Hedrick, F. Gao, and H. Zhang, "Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 3, pp. 46–58, Fall. 2017.

[27] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 899–910, May 2017.

[28] P. Liu, A. Kurt, and U. Ozguner, "Distributed model predictive control for cooperative and flexible vehicle platooning," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 3, pp. 1115–1128, May 2019.

[29] (2024). *Global CVT Transmission Market 2024–2030*. [Online]. Available: https://www.mobilityforesights.com

[30] V. Turri, B. Besselink, and K. H. Johansson, "Gear management for fuel-efficient heavy-duty vehicle platooning," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Las Vegas, NV, USA, Dec. 2016, pp. 1687–1694.

[31] G. Li and D. Görges, "Ecological adaptive cruise control for vehicles with step-gear transmission based on reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4895–4905, Nov. 2020.

[32] Y. Shao and Z. Sun, "Vehicle speed and gear position co-optimization for energy-efficient connected and autonomous vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 4, pp. 1721–1732, Jul. 2021.

[33] S. Mallick, G. Battocletti, Q. Dong, A. Dabiri, and B. De Schutter, "Learning-based MPC for fuel efficient control of autonomous vehicles with discrete gear selection," *IEEE Control Syst. Lett.*, vol. 9, pp. 1117–1122, 2025.

[34] H. Zhang, L. Du, and J. Shen, "Hybrid MPC system for platoon based cooperative lane change control using machine learning aided distributed optimization," *Transp. Res. B, Methodol.*, vol. 159, pp. 104–142, May 2022.

[35] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, Jul. 2001.

[36] J. Lan, D. Zhao, and D. Tian, "Data-driven robust predictive control for mixed vehicle platoons using noisy measurement," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 6, pp. 6586–6596, Jun. 2023.

[37] M. Doostmohammadian, A. Aghasi, and H. R. Rabiee, "Notice of removal: Fully distributed and quantized algorithm for MPC-based autonomous vehicle platooning optimization," in *Proc. 12th RSI Int. Conf. Robot. Mechatronics (ICRoM)*, Dec. 2024, pp. 051–056.

[38] H. Liu, D. Chu, W. Zhong, B. Gao, Y. Lu, S. Han, and W. Lei, "Compensation control of commercial vehicle platoon considering communication delay and response lag," *Comput. Electr. Eng.*, vol. 119, Nov. 2024, Art. no. 109623.

[39] J. Zhan, L. Zhang, and Y. Chen, "Asynchronous platoon control for connected vehicles with intermittent and delayed information transmission," *IEEE Trans. Autom. Control*, vol. 68, no. 11, pp. 6875–6882, Nov. 2023.

[40] M. Doostmohammadian, U. A. Khan, and N. Meskin, "On the redundant distributed observability of mixed traffic transportation systems," *EURASIP J. Adv. Signal Process.*, vol. 2025, no. 1, p. 61, 2025.

[41] W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: Stability and string stability," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 620–633, Mar. 2012.

[42] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: Predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, Jul. 2001.

[43] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optim.*, vol. 19, pp. 79–102, Feb. 2016.

[44] A. N. Venkat, I. A. Hiskens, J. B. Rawlings, and S. J. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 6, pp. 1192–1206, Apr. 2008.

[45] D. E. Bernal, Q. Chen, F. Gong, and I. E. Grossmann, "Mixed-integer nonlinear decomposition toolbox for pyomo (MindtPy)," *Comput. Aided Chem. Eng.*, vol. 44, pp. 895–900, Jan. 2018.

SAMUEL MALLICK received the B.Sc. and M.Sc. degrees from The University of Melbourne, in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree with Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

His research interests include model predictive control, reinforcement learning, and distributed control of large-scale and hybrid systems.

AZITA DABIRI received the Ph.D. degree from the Automatic Control Group, Chalmers University of Technology, in 2016. She was a Postdoctoral Researcher with the Department of Transport and Planning, TU Delft, from 2017 to 2019. In 2019, she received an ERCIM Fellowship and also a Marie Curie Individual Fellowship, which allowed her to perform research with Norwegian University of Technology (NTNU) as a Postdoctoral Researcher, from 2019 to 2020, before joining Delft Center for Systems and Control, TU Delft, as an Assistant Professor. Her research interests include the integration of model-based and learning-based control and its applications in transportation networks.

BART DE SCHUTTER (Fellow, IEEE) received the Ph.D. degree (summa cum laude) in applied sciences from KU Leuven, Belgium, in 1996.

He is currently a Full Professor and the Head of the Department with Delft Center for Systems and Control, Delft University of Technology, The Netherlands. His research interests include multi-level and multi-agent control, model predictive control, learning-based control, and control of hybrid systems, with applications in intelligent transportation systems and smart energy systems. He is a Senior Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL.