Experimental validation of a semi-distributed sequential quadratic programming method for optimal coordination of automated vehicles at

Received: 3 September 2018 | Revised: 13 August 2019 | Accepted: 10 February 2020

DOI: 10.1002/oca.2592

RESEARCH ARTICLE

WILEY

Experimental validation of a semi-distributed sequential quadratic programming method for optimal coordination of automated vehicles at intersections

Robert Hult[1] | **Mario Zanon**[2] | **Gianluca Frison**[3] | **Sébastien Gros**[1,4] | **Paolo Falcone**[1]

[1]Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

[2]IMT School of Advanced Studies, Lucca, Italy

[3]Department of Microsystems Engineering, University of Freiburg, Freiburg, Germany

[4]Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

Correspondence
Robert Hult, Department of Electrical Engineering, Chalmers University of Technology, Hörsalsvägen 11, Gothenburg SE-41296, Sweden.
Email: robert.hult@chalmers.se

Summary

In this article, we study the optimal coordination of automated vehicles at intersections. The problem can be stated as an optimal control problem (OCP), which can be decomposed as a bi-level scheme composed by one nonlinear program (NLP) which schedules the access to the intersection and one OCP per vehicle which computes the appropriate vehicle commands. We discuss a practical implementation of the bi-level controller where the NLP is solved with a tailored semi-distributed sequential quadratic programming (SQP) algorithm that enables distribution of most computation to the vehicles. Results from an extensive experimental campaign are presented, where the bi-level controller and the semi-distributed SQP are implemented on a test setup consisting of three automated vehicles. In particular, we show that the vehicle-level controller can enforce the scheduled intersection access beyond the accuracy admitted by the sensor system, and that the bi-level controller can handle large perturbations and large communication delays, which makes the scheme applicable in practical scenarios. Finally, the use of wireless communication introduces delays in the outer control loop. To allow faster feedback, we introduce a real-time iteration (RTI) like variation of the bi-level controller. Experimental and simulated results indicate that the RTI-like variation offers comparable performance using less computation and communication.

KEYWORDS

automated vehicles, distributed model predictive control, distributed nonlinear programming, intersection coordination

1 | INTRODUCTION

The current trend toward automation of road vehicles can be expected to continue, and eventually most vehicles will be fully automated and communicating. This technology can be leveraged to obtain synergistic effects through cooperation between the automated vehicles, and thereby enable drastic improvements to the traffic system. In this article, we discuss an algorithm necessary for one such improvement: the automation of intersection crossings. With all vehicles automated, communicating and cooperative, the traffic-lights, signs and rules used today could be removed and the vehicles could

instead rely on automated coordination controllers. As discussed in Reference 1, the potential benefits include increased safety, increased energy efficiency and higher traffic throughput.

However, there are several challenges that must be addressed before coordination algorithms can be applied in practice. Most importantly, such controllers must be able to guarantee that no collisions occur, and in particular, the guarantees must be applicable to scenarios with uncertainty. This includes handling unexpected events and the online recoordination of vehicles in the presence of new information. Furthermore, a useful coordination algorithm must be scalable to be relevant for more than small scenarios. However, since finding the optimal collision free motion profiles for vehicles crossing an intersection is a combinatorial problem, there are computational scalability issues. In fact, determining the existence of even one collision free solution has been shown to be an NP-hard problem in the general case.[2] Moreover, it is a known problem that vehicle-to-vehicle (V2V) communication systems have capacity limits.[3] A practically useful coordination algorithm must therefore also scale well in terms of both how often communication is required and the data volumes involved.

Although the application of intersection coordination algorithms lies in the future, a number of contributions have been made during the last decade, many of which are surveyed in References 4 and 5. In the literature, various heuristics are used to address the challenges of the problem: the motion profile is typically the result of a rule-based controller which switches between discrete behavioral modes,[6-9] or is obtained from a restricted space, for example, trapezoidal[10] or linear[11] velocity profiles. On the other hand, a number of approaches based on optimal control (OC) formulations of the coordination problem can be found in the literature, for example, References 12-22 and 23. In most cases, the selection of a crossing order is separated from the computation of the optimal state and control trajectories to avoid the combinatorial complexity of the solution space. For instance, in References 14,16,19 variations of "First-Come-First-Served" policies are first used to produce a crossing order and OC problems constrained to satisfy this order are then solved. Using a similar strategy,[15] leverages results from polling-systems to compute the crossing order, while in References 24 and 23 mixed-integer quadratic programming (QP) is used to compute an approximately optimal crossing order. A different approach is taken in Reference 12, where a heuristic gives a decision order rather than a crossing order. The vehicles thereafter sequentially solve optimal control problems (OCPs), where each vehicle is constrained to avoid collisions with the vehicles that precedes it in the decision order. The application of OC formulations to closed-loop control is considered in References 13,21 and 25.

As discussed in Reference 1, the benefits of OC approaches in general include the ability to consider a wider range of applicable motion profiles and include constraints. Given the severity of collisions, closed-loop control, that is, the recalculation of control commands based on measurements of the system state, is a necessity to handle the uncertainty that is present in real scenarios. In many cases, OC schemes can leverage well established theory to derive properties of closed-loop control schemes, account for various forms of uncertainty and construct efficient solution algorithms.

In this article, we use the OC formulation of the coordination problem first presented in Reference 24 but focus on finding the optimal solution for a given crossing order. With this formulation, the problem is given a hierarchical structure, where optimal, collision free intersection occupancy time-slots are obtained as the solution to a nonlinear program (NLP), and the optimal state and control trajectories as the solution to OCPs that are separable between the vehicles. This structure enables a bi-level model predictive control (MPC) architecture where coordination is separated from vehicle control. In particular, the outer, *intersection*-level, controller computes and updates optimal, nonoverlapping time-slots based on the current vehicle states, and the lower, *vehicle*-level, controllers compute the control commands for the vehicles, given a time-slot and the current state.

In earlier work, we proposed a semi-distributed sequential quadratic programming (SQP) approach for the solution of the time-slot NLP.[26] The algorithm was extended in Reference 27 where a convergence proof also was given. We established the persistent feasibility of the bi-level MPC scheme and discussed robustness aspects in Reference 25, and presented experimental results. Extensions to economic nonlinear MPC were presented in Reference 22, and a comparison of References 26 and 27 was given in Reference 28, supported by experimental data.

The SQP procedure of References 26 and 27 has the property that most computations can be performed on-board the vehicles and the algorithm's internal message passing can be performed using V2V communication. Consequently, the SQP procedure can be used to close the intersection-level control loop (repeatedly solve the time-slot NLP online) in a semi-distributed manner. While such a scheme has several desirable properties, it is necessary to evaluate its usefulness in a practical setting. First, the algorithmic performance needs to be assessed for real scenarios where, in particular, the effects of delays inherent to the use of real communication systems must be studied. Second, the effects of both algorithmic performance and real-world perturbations on the performance of the bi-level controller must be investigated and possible issues addressed. To this end, we describe a practical implementation of the bi-level controller in this

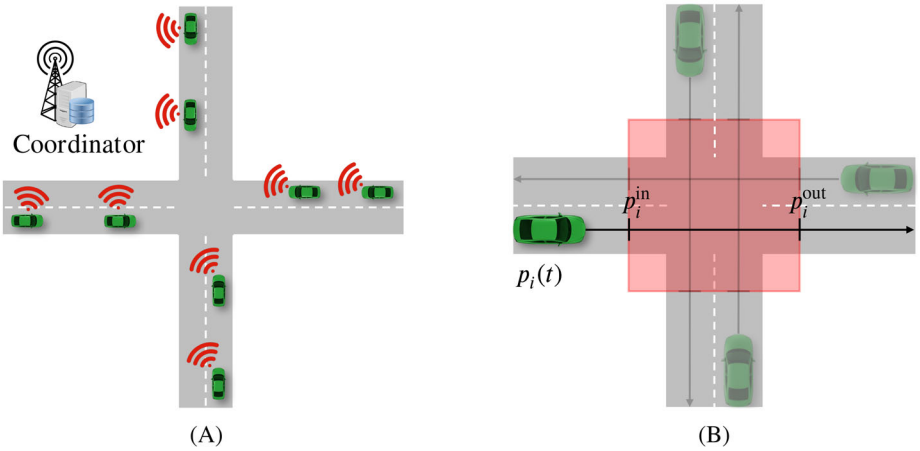


FIGURE 1 A, Contains a schematic illustration of the scenarios considered in this article. B, Illustrates how the intersection is modeled: the arrows show the fixed paths of the vehicles, and the red square illustrates the zone inside the intersection where collisions can occur [Colour figure can be viewed at wileyonlinelibrary.com]

article, where the intersection-level control loop is closed using the SQP presented in References 26,27. In particular, we detail the application of the bi-level controller on a test setup consisting of three automated vehicles, where the SQP is solved in a semi-distributed fashion using V2V communication, and the vehicle-level control loop is closed using the high-performance QP solver HPMPC.[29] Furthermore, we introduce modifications to the bi-level controller that increase the practical applicability of the scheme. First, a relaxation of the vehicle-level MPC problem is presented, which resolves infeasibility issues inherent to the formulation in References 26,27. Second, two modifications to the intersection-level controller are introduced to handle the large computational delays that can arise due to the execution of the SQPs over a wireless network which introduces delays. In particular, we propose a scheme where the intersection-level control loop is closed in a real-time iteration (RTI) like fashion[30] to allow faster feedback. That is, instead of solving the NLP to convergence, the intersection-level control loop consists of the time-slot updates resulting from one SQP iteration. Moreover, we present results from an extensive experimental campaign where the implementation was evaluated. We discuss the algorithmic performance and provide a detailed study of the SQP execution times, where the experimental data is compared with ideal cases. Comparative data is provided for both experimental and simulated cases, where the system is subject to both large and small perturbations.

The remainder of the article is organized as follows: The modeling and OC formulation of the problem are introduced in Section 2, while the semi-distributed SQP and practical implementation are discussed in Sections 3 and 4, respectively. The experimental results are presented in Section 5 and the article is concluded with a discussion in Section 6.

2 | PROBLEM FORMULATION

In this section, we introduce the modeling and OC formulation of the intersection problem and discuss how the formulation can be decomposed which enables a bi-level structure for closed-loop control. The computational and other practical aspects, for example, what computation is performed where, is discussed in Section 3.

2.1 | Modeling

We consider problems such as that shown in Figure 1A, where N_a vehicles approach an intersection equipped with a central coordinating unit. We assume that all involved vehicles are automated, cooperative and participate in the coordination procedure and that no noncooperative entities (eg, pedestrians and bicyclists) are present. For simplicity, we also assume that no vehicles makes turns or change lanes, but note that such vehicles could be included using the methods discussed in Reference 31.

We assume that the vehicles move along predefined paths and that the vehicle dynamics along the paths can be described by

$$\dot{x}_i(t) = A_i^c x_i(t) + B_i^c u_i(t), \tag{1}$$

where $x_i(t) \in \mathbb{R}^n$, $u_i(t) \in \mathbb{R}^m$ and $A_i^c \in \mathbb{R}^{n\times n}$, $B_i^c \in \mathbb{R}^{n\times m}$. Specifically, the state vector is such that $x_i(t) = (p_i(t), y_i(t))$, where $p_i(t) \in \mathbb{R}$ is the position of the center of the vehicle on its path and $y_i(t) \in \mathbb{R}^{n-1}$ collects all non position states (eg, velocity and acceleration). Moreover, the vehicle state and control trajectories are subject to constraints of the form

$$D_i x_i(t) + G_i u_i(t) \geq b_i, \tag{2}$$

capturing, for example, actuation limitations and passenger comfort restrictions.

We consider only D_i, G_i, A_i^c, B_i^c such that $\dot{p}(t) \geq 0$, that is, the dynamics and constraints are such that no vehicle can reverse.

As shown in Figure 1, we define the intersection as an interval $[p_i^{\text{in}}, p_i^{\text{out}}]$ on the path of each vehicle such that collisions are avoided if $p_i(t) \in [p_i^{\text{in}}, p_i^{\text{out}}] \Rightarrow p_j(t) \notin [p_j^{\text{in}}, p_j^{\text{out}}]$ hold, for all vehicles $i \neq j$ *. Furthermore, we define the time-of-entry, t_i^{in}, and time-of-clearance, t_i^{out}, of the intersection through,

$$p_i(t_i^{\text{in}}) = p_i^{\text{in}} \text{ and } p_i(t_i^{\text{out}}) = p_i^{\text{out}}, \tag{3}$$

respectively. Collision avoidance is thereby ensured if

$$t_i^{\text{out}} \leq t_j^{\text{in}} \tag{4}$$

for all vehicle pairs (i, j) such that vehicle i crosses the intersection before vehicle j. In the remainder of the article, we denote $T_i = (t_i^{\text{in}}, t_i^{\text{out}})$ the *time-slot* of vehicle i, and state that a vehicle conforms to T_i if it only occupies the intersection within $[t_i^{\text{in}}, t_i^{\text{out}}]$.

2.2 | OC formulation

With the objective

$$J_i(x_i(t), u_i(t)) = V_{i,f}(x_i(t_f)) + \int_0^{t_f} \ell_i(x_i(t), u_i(t))\mathrm{d}t, \tag{5}$$

where $V_{i,f}(x_i(t_f))$ and $\ell_i(x_i(t), u_i(t))$ are convex and quadratic, and t_f is fixed, we state problem of optimally coordinating the vehicles through the intersection as

$$\min_{\boldsymbol{T}, x(t), u(t)} \quad \sum_{i=1}^{N_a} J_i(x_i(t), u_i(t)) \tag{6a}$$

$$\text{s.t.} \quad x_i(0) = \hat{x}_{i,0}, \qquad i \in \mathbb{I}_{[1,N_a]}, \tag{6b}$$

$$\dot{x}_i(t) = A_i^c x_i(t) + B_i^c u_i(t), \qquad i \in \mathbb{I}_{[1,N_a]}, \tag{6c}$$

$$D_i x_i(t) + G_i u_i(t) \geq b_i, \qquad i \in \mathbb{I}_{[1,N_a]}, \tag{6d}$$

$$p_i(t_i^{\text{in}}) = p_i^{\text{in}}, \quad p_i(t_i^{\text{out}}) = p_i^{\text{out}}, \qquad i \in \mathbb{I}_{[1,N_a]}, \tag{6e}$$

$$t_i^{\text{out}} \leq t_{i+1}^{\text{in}} \qquad i \in \mathbb{I}_{[1,N_a-1]}. \tag{6f}$$

where the vehicles are ordered such that vehicle i crosses before vehicle $i+1$. Here, $\mathbb{I}_{[a,b]} = \{a, \dots, b\}$ for integers $a < b$, $x(t) = (x_1(t), \dots, x_{N_a}(t))$, $u(t) = (u_1(t), \dots, u_{N_a}(t))$, $\boldsymbol{T} = (T_1, \dots, T_{N_a})$ and $\hat{x}_{i,0}$ is the initial state of vehicle i.

*We want to emphasize that the definition of the intersection easily can be subdivided into several mutual-exclusion zones, each with its own start and stop position, as done in Reference 23. However, for simplicity of presentation, the article is developed with the single zone shown in Figure 1.

2.2.1 | Decomposition

It was shown in Reference 24 that the coordination problem can be decomposed in a hierarchical fashion, where the time-slot schedule $\boldsymbol{T}$ is the solution of a NLP, and the vehicle state and control trajectories $x_i(t), u_i(t)$ are the solution to separable vehicle OCPs. The following NLP computes the optimal time-slot schedule $\boldsymbol{T}$ for a given order S

$$\min_{\boldsymbol{T}} \quad \sum_{i=1}^{N_a} V_i(\hat{x}_{i,0}, T_i) \tag{7a}$$

$$\text{s.t.} \quad T_i \in \text{domain}(V_i(\hat{x}_{i,0}, T_i)), \qquad i \in \mathbb{I}_{[1,N_a]}, \tag{7b}$$

$$t_i^{\text{out}} \leq t_{i+1}^{\text{in}} \qquad i \in \mathbb{I}_{[1,N_a-1]}, \tag{7c}$$

where $V_i(\hat{x}_{i,0}, T_i)$ is defined as the optimal value function of the OCP of vehicle i

$$V_i(\hat{x}_{i,0}, T_i) = \min_{x_i(t),u_i(t)} \quad J_i(x_i(t), u_i(t)) \tag{8a}$$

$$\text{s.t.} \quad x_i(0) = \hat{x}_{i,0}, \tag{8b}$$

$$\dot{x}_i(t) = A_i^c x_i(t) + B_i^c u_i(t), \tag{8c}$$

$$D_i x_i(t) + G_i u_i(t) \geq b_i, \tag{8d}$$

$$p_i(t_i^{\text{in}}) = p_i^{\text{in}}, \quad p_i(t_i^{\text{out}}) = p_i^{\text{out}}, \tag{8e}$$

For the optimal T_i, (8) gives the optimal state and control trajectories $x_i(t), u_i(t)$.

2.2.2 | Discretization

For practical reasons, we consider piecewise constant inputs and discretize the vehicle dynamics using the sampling time t_s when we solve the vehicle problem (8). More precisely, we define $x_{i,k} = x_i(t_k)$ and $u(t) = u_{i,k}, \forall t \in [t_k, t_{k+1}[$, with $t_k = kt_s$, and the state update function $x_{i,k+1} = A_i x_{i,k} + B_i u_{i,k}$, where $A_i = \mathcal{A}_i(t_s) = \exp(A_i^c t_s)$, $B_i = \mathcal{B}_i(t_s) = \int_0^{t_s} \exp(A_i^c(t_s - s))B_i^c ds$ and where $x_{i,k} = (p_{i,k}, v_{i,k})$.

Since the discrete position is defined only at t_k, (3) defines values of t_i^{in} and t_i^{out} that are integer multiples of t_s in the discrete time case. To allow t_i^{in} and t_i^{out} to assume continuous values, we define a continuous time representation of the position using the discrete time state and control sequences as

$$p_i^d(t, w_i) = [1, \mathbf{0}^{n-1}] \left(\mathcal{A}_i(t - t_k)x_{i,k} + \mathcal{B}_i(t - t_k)u_{i,k} \right), \quad k = \text{floor}(t/t_s), \tag{9}$$

where $w_i = (x_{i,0}, u_{i,0}, \ldots, x_{i,N-1}, u_{i,N-1}, x_{i,N})$ and where $t_f = Nt_s$. The discrete time statement of the objective function is

$$J_i^d(w_i) = x_{i,N}^\top P_i x_{i,N} + \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_{i,k} \\ u_{i,k} \end{bmatrix}^\top Q_i \begin{bmatrix} x_{i,k} \\ u_{i,k} \end{bmatrix} + q_i^\top \begin{bmatrix} x_{i,k} \\ u_{i,k} \end{bmatrix} \tag{10}$$

whereby we have the discrete time formulation of the OCP (8) of each vehicle i as

$$V_i(\hat{x}_{i,0}, T_i) = \min_{w_i} \quad J_i^d(w_i) \tag{11a}$$

$$\text{s.t.} \quad x_{i,0} = \hat{x}_{i,0}, \tag{11b}$$

$$x_{i,k+1} = A_i x_{i,k} + B_i u_{i,k}, \quad k \in \mathbb{I}_{[0,N-1]}, \tag{11c}$$

$$D_i x_{i,k} + G_i u_{i,k}(t) \geq b_i, \quad k \in \mathbb{I}_{[0,N-1]}, \tag{11d}$$

$$p_i^d(t_i^{\text{in}}, w_i) = p_i^{\text{in}}, \tag{11e}$$

$$p_i^d(t_i^{\text{out}}, w_i) = p_i^{\text{out}}, \tag{11f}$$

which can be solved for real valued t_i^{in} and t_i^{out} in $[0, Nt_s]$. We note here that for w_i which satisfies (11c), $p_i^d(t, w_i)$ is K-times continuously differentiable, where K is the relative degree of (1) with the position $p_i(t)$ as the output.[26]

2.2.3 | Problem properties

The constraint set domain$(V_i(\hat{x}_{i,0}, T_i))$ in (7b) is implicitly defined as the set of T_i for which the optimization problem (11) is feasible given the initial state $\hat{x}_{i,0}$. However, it was shown in Reference 24 that domain$(V_i(\hat{x}_{i,0}, T_i))$ can be written as

$$h_i(\hat{x}_{i,0}, T_i) = \begin{bmatrix} L^{\text{in}}(\hat{x}_{i,0}) - t_i^{\text{in}} \\ t_i^{\text{in}} - U^{\text{in}}(\hat{x}_{i,0}) \\ U(\hat{x}_{i,0}, t_i^{\text{in}}) - t_i^{\text{out}} \\ t_i^{\text{out}} - L(\hat{x}_{i,0}, t_i^{\text{in}}) \end{bmatrix} \geq 0, \tag{12}$$

where $L^{\text{in}}(\hat{x}_{i,0})$, $U^{\text{in}}(\hat{x}_{i,0})$, $U(\hat{x}_{i,0}, t_i^{\text{in}})$, and $L(\hat{x}_{i,0}, t_i^{\text{in}})$ are defined as the solutions to the NLPs

$$L^{\text{in}}(\hat{x}_{i,0}) = \min_{w_i, t} t \quad \text{s.t.} \quad (11b),\ (11c),\ (11d),\ p_i^d(t, w_i) = p_i^{\text{in}}, \tag{13}$$

$$U^{\text{in}}(\hat{x}_{i,0}) = \max_{w_i, t} t \quad \text{s.t.} \quad (11v),\ (11c),\ (11d),\ p_i^d(t, w_i) = p_i^{\text{in}}, \tag{14}$$

$$L(\hat{x}_{i,0}, t_i^{\text{in}}) = \min_{w_i, t} t \quad \text{s.t.} \quad (11b),\ (11c),\ (11d),\ p_i^d(t_i^{\text{in}}, w_i) = p_i^{\text{in}},\ p_i^d(t, w_i) = p_i^{\text{out}}, \tag{15}$$

$$U(\hat{x}_{i,0}, t_i^{\text{in}}) = \max_{w_i, t} t \quad \text{s.t.} \quad (11b),\ (11c),\ (11d),\ p_i^d(t_i^{\text{in}}, w_i) = p_i^{\text{in}},\ p_i^d(t, w_i) = p_i^{\text{out}}. \tag{16}$$

That is, t_i^{in} must lie between the earliest and latest time-of-entry that the vehicle can perform. Similarly, for a specified time-of-entry t_i^{in}, t_i^{out} must lie between the earliest and latest time-of-clearance the vehicle can perform. Moreover, it was shown in Reference 26 that if a mild technical assumption holds, the optimal solutions to the linear programs (LPs)

$$w_i^{\text{ub}}(\hat{x}_{i,0}, t_i^{\text{in}}) = \arg\min_{w_i} p_{i,N} \quad \text{s.t.} \quad (11b),\ (11c),\ (11d),\ p_i^d(t_i^{\text{in}}, w_i) = p_i^{\text{in}}, \tag{17}$$

$$w_i^{\text{lb}}(\hat{x}_{i,0}, t_i^{\text{in}}) = \arg\max_{w_i} p_{i,N} \quad \text{s.t.} \quad (11b),\ (11c),\ (11d),\ p_i^d(t_i^{\text{in}}, w_i) = p_i^{\text{in}}, \tag{18}$$

are also solutions to (15) and (16), respectively. Consequently, (15) and (16) can be evaluated by first solving the LPs (17), (18) and thereafter solving

$$p_i^d\left(U(\hat{x}_{i,0}, t_i^{\text{in}}), w_i^{\text{ub}}(\hat{x}_{i,0}, t_i^{\text{in}})\right) - p_i^{\text{out}} = 0, \tag{19}$$

$$p_i^d\left(L(\hat{x}_{i,0}, t_i^{\text{in}}), w_i^{\text{lb}}(\hat{x}_{i,0}, t_i^{\text{in}})\right) - p_i^{\text{out}} = 0 \tag{20}$$

for $U(\hat{x}_{i,0}, t_i^{\text{in}})$ and $L(\hat{x}_{i,0}, t_i^{\text{in}})$. The bounds $L^{\text{in}}(\hat{x}_{i,0})$ and $U^{\text{in}}(\hat{x}_{i,0})$ can be obtained similarly. For notational convenience, in the remainder of the article we will only include the explicit dependence on $\hat{x}_{i,0}$ in $h_i(\hat{x}_{i,0}, T_i)$ and $V_i(\hat{x}_{i,0}, T_i)$ when necessary.

2.3 | Receding horizon implementation

In order to reject perturbations and compensate for model inaccuracies, the solution to the optimal coordination problem can be applied in a receding horizon fashion in a MPC. In particular, the decomposed formulation offers a natural separation between coordination and vehicle control. This enables a bi-level control structure: an outer, intersection-level, control loop computes collision free time-slots by solving (7) at the current state, while inner, vehicle-level, control loops

compute the vehicle control command $u_{i,0}$ for a given time-slot T_i at the current state through (11). Feedback is thereby due to estimates of the current state in the inner control loops, and the cost function $V_i(\hat{x}_{i,0}, T_i)$ and the constraint set $h_i(\hat{x}_{i,0}, T_i)$ in the outer control loop. This scheme has the benefit that perturbations acting on one vehicle are counteracted by all vehicles, that is, the optimal time-slot of vehicle i at time k, $T_i(X_k)$, is a function of the state of all vehicles $X_k = (x_{1,k}, \dots, x_{N_a,k})$ through (7).

In principle, the time-slot schedule could be computed once and the rejection of possible perturbations handled by the inner control loops. However, by closing the outer control loop the system can (a) reject larger perturbations by adjusting the time-slot schedule $\boldsymbol{T}$ and thereby provide collision avoidance in more demanding scenarios and (b) continuously improve the solution. The bi-level controller is discussed further in Reference 25, where the closed-loop system is shown to be persistently feasible and stable.

The computations required to close the inner, vehicle-level control loop (solution of QP (11)) can be performed independently on each vehicle. However, closing the outer, intersection-level control loop involves the solution of NLP (7), which requires information from all vehicles. In the following section, we discuss an algorithm which solves NLP (7), where most computations are distributed and performed on-board the vehicles.

3 | A SEMI-DISTRIBUTED SQP METHOD

Considering the intended application, an algorithm where much of the computations required to solve NLP (7) can be performed on board the vehicles is desirable as it improves scalability. However, if computations are performed on board the vehicles, the algorithm requires information exchange over the V2V network. As is reported in Reference 3, there are scalability issues with the current V2V technology and frequent and large data exchange should be avoided. Consequently, second-order optimization methods are preferable to first-order ones, as the former in general need fewer iterations to find a solution to the problem. For this reason, a semi-distributed SQP algorithm was proposed in References 26,27. We recall the details of the SQP algorithm in Section 3.1, discuss its application in a practical setting with semi-distributed computation in Section 3.2, and present two different approaches to its application for closed-loop control Section 3.3.

3.1 | Sequential quadratic programming

Using the developments of Section 2.2.3, we rewrite NLP (7) as

$$\min_{\boldsymbol{T}} \; V(\boldsymbol{T}) \quad \text{s.t.} \;\; h(\boldsymbol{T}) \geq 0, \tag{21}$$

where $V(\boldsymbol{T}) = \sum_{i=1}^{N_a} V_i(T_i)$ and we lumped constraints (7b)-(7c) in the function $h(\boldsymbol{T})$. The associated Lagrangian function is defined as $\mathcal{L}(\boldsymbol{T}, \mu) := V(\boldsymbol{T}) - \mu^\top h(\boldsymbol{T})$, where $\mu = (\mu_1, \dots, \mu_{N_a}, \mu_s)$. Here, μ_i are the Lagrange multipliers of the constraint $h_i(T_i) \geq 0$ and μ_s the multipliers of the precedence constraints (7c), which we write as $h_s(\boldsymbol{T}) \geq 0$

Starting from an initial guess $z^{(0)} = (\boldsymbol{T}^{(0)}, \mu^{(0)})$, SQP iteratively updates the primal-dual solution candidate $z^{(c)}$ using

$$z^{(c+1)} = z^{(c)} + \alpha^{(c)} \Delta z^{(c)}, \tag{22}$$

with $\alpha^{(c)} \in (0, 1]$ and $\Delta z^{(c)} = (\Delta \boldsymbol{T}^{(c)}, \; \tilde{\mu}^{(c)} - \mu^{(c)})$. Here, $(\Delta \boldsymbol{T}^{(c)}, \; \tilde{\mu}^{(c)})$ is the primal-dual solution of the QP subproblem

$$\min_{\Delta \boldsymbol{T}} \; \frac{1}{2} \Delta \boldsymbol{T}^\top H^{(c)} \Delta \boldsymbol{T} + \nabla_{\boldsymbol{T}} V(\boldsymbol{T}^{(c)})^\top \Delta \boldsymbol{T} \tag{23a}$$

$$\text{s.t.} \;\; h(\boldsymbol{T}^{(c)}) + \nabla_{\boldsymbol{T}} h(\boldsymbol{T}^{(c)})^\top \Delta \boldsymbol{T} \geq 0, \tag{23b}$$

where $H^{(c)}$ is a positive-definite approximation of the Lagrange function Hessian $\nabla^2_{\boldsymbol{T}} \mathcal{L}(\boldsymbol{T}^{(c)}, \mu^{(c)})$. Variants of SQP differ primarily in the computations of the step size $\alpha^{(k)}$ and the Lagrangian Hessian approximation $H^{(k)}$. We describe next the methods employed to solve (7). For more details on SQP see, for example, Reference 32.

3.1.1 | Hessian approximation

To ensure that the QP subproblems (23) are convex, it is required that the reduced Hessian is positive-definite. While there are several ways of enforcing positive-definiteness of the reduced Hessian, we adopt the strategy of adding enough curvature in all negative-curvature directions to ensure that the full Hessian is positive-definite. In particular, we note that NLP (21) is such that the Hessian has the block-diagonal form

$$\nabla^2_{\boldsymbol{T}}\mathcal{L}(\boldsymbol{T}^{(c)},\mu^{(c)}) = \operatorname{diag}(L_i(T_1^{(c)},\mu_1^{(c)}),\dots,L_{N_a}(T_{N_a}^{(c)},\mu_{N_a}^{(c)})), \tag{24}$$

where $L_i(T_i^{(c)},\mu_i^{(c)}) = \nabla^2_{T_i}V_i(T_i^{(c)}) + \langle\mu_i^{(c)},\nabla^2_{T_i}h_i(T_i^{(c)})\rangle$. We define a positive-definite approximation to $\nabla^2_{\boldsymbol{T}}\mathcal{L}(\boldsymbol{T}^{(c)},\mu^{(c)})$ as $H_i^{(c)} = \operatorname{diag}(H_1^{(c)},\dots,H_{N_a}^{(c)})$, where $H_i^{(c)} = E_i^{(c)}D_i^{(c)}{E_i^{(c)}}^{\top}$. Here, $D_i^{(c)}$ is a diagonal matrix where the jth diagonal element is $d_{i,j}^{(c)} = \max(e_{i,j}^{(c)},\varepsilon)$, where $e_{i,j}$ is the jth eigenvalue of $L_i(T_i^{(c)},\mu_i^{(c)})$ and $\varepsilon > 0$ is a constant. The columns of $E_i^{(c)}$ are the normalized eigenvectors corresponding to the eigenvalues in D_i. The required eigenvalue decomposition is cheap due to the small size of the blocks $L_i(T_i^{(c)},\mu_i^{(c)})$.

3.1.2 | Step size selection

In order to guarantee convergence of SQP algorithms, the step-size $\alpha^{(c)}$ must be selected such that progress toward a solution to the problem is made. In this article, we employ a line search on the so-called ℓ_1 merit function, which is defined as

$$M(\boldsymbol{T}^{(c)}) = V(\boldsymbol{T}^{(c)}) + \rho^{(c)}\|h^-(\boldsymbol{T}^{(c)})\|_1, \tag{25}$$

where $h^-(\boldsymbol{T}^{(c)}) = \min(h(\boldsymbol{T}^{(c)}),0)$ and $\rho^{(c)}$ is a parameter chosen so that $\rho^{(c)} > \|\mu^{(c)}\|_1$. Progress toward a solution is ensured when $\alpha^{(c)}$ is selected such that the Armijo condition is satisfied:

$$M(\boldsymbol{T}^{(c)} + \alpha^{(c)}\Delta\boldsymbol{T}^{(c)}) \le M(\boldsymbol{T}^{(c)}) + \gamma D_{\Delta\boldsymbol{T}^{(c)}}M(\boldsymbol{T}^{(c)})\alpha^{(c)}, \tag{26}$$

where $\gamma \in (0,0.5]$ and $D_{\Delta\boldsymbol{T}^{(c)}}M(\boldsymbol{T}^{(c)})$ is the derivative of $M(\boldsymbol{T})$ in the direction of $\Delta\boldsymbol{T}^{(c)}$, evaluated at $\boldsymbol{T}^{(c)}$. Provided that $\Delta\boldsymbol{T}^{(c)}$ is a descent direction on (25), $\alpha^{(c)}$ which satisfies (26) exists and can be found by so-called backtracking, that is, by successively decreasing $\alpha^{(c)}$ from 1 until (26) is satisfied.[32]

Since the constraint $h_i(T_i) \ge 0$ defines the set of feasible parameters for the parametric QP (11), and $V_i(T_i)$ is the optimal value function for the same QP, we note that $V_i(T_i)$, and thereby $M(\boldsymbol{T})$, is undefined when $h_i(T_i) \not\ge 0$. We resolve this issue by using the projection-based method of Reference 27, where the merit function is evaluated at $\mathcal{P}(T_i^{(c)} + \alpha^{(c)}\Delta T_i^{(c)})$ rather than at $T_i^{(c)} + \alpha^{(c)}\Delta T_i^{(c)}$. Here, $\mathcal{P}(T_i)$ is a projection operator, defined as

$$\mathcal{P}(T_i) := \left(t_i^{\text{in}},\ \min\left(U(t_i^{\text{in}}),\max\left(L(t_i^{\text{in}}),t_i^{\text{out}}\right)\right)\right). \tag{27}$$

It was shown in Reference 27 that if $\Delta\boldsymbol{T}^{(c)}$ is a descent direction on (25), a small enough $\alpha^{(c)}$ exists which satisfies (26), such that one can perform backtracking based on $M(\mathcal{P}(\boldsymbol{T}^{(c)} + \alpha^{(c)}\Delta\boldsymbol{T}^{(c)}))$. Note that with this modification, the ℓ_1 merit function reads $M(\mathcal{P}(\boldsymbol{T})) = V(\mathcal{P}(\boldsymbol{T})) + \rho\|h_s^-(\mathcal{P}(\boldsymbol{T}))\|$ and the primal-dual update is

$$T_i^{(c+1)} = \mathcal{P}\left(T_i^{(c)} + \alpha^{(c)}\Delta T_i^{(c)}\right), \quad i \in \mathbb{I}_{[1,N_a]} \tag{28a}$$

$$\mu^{(c+1)} = \mu^{(c)} + \alpha^{(c)}\Delta\mu^{(c)}. \tag{28b}$$

An alternative solution to the issue of nondefined $M(\boldsymbol{T})$ is to soften the position constraints (11e) and (11f) with an ℓ_1 penalty as suggested in Reference 26. However, in doing so the objective in the quadratic subproblem (42) will be

dominated by the penalty term whenever it is evaluated at $h_i(T_i) \not\geq 0$ for one vehicle, and a sharp nonsmoothness appears at points where the problem becomes feasible. The algorithmic performance of this method has been found to be worse than the method based on (27).[28]

3.1.3 | Calculation of derivatives

The first and second-order derivatives of the objective function components $V_i(T_i^{(c)})$ and constraint components $h_i(T_i^{(c)})$ are required to form the QP-subproblem (23). Since $V_i(T_i^{(c)})$ is the optimal value function of the QP (11) and $h_i(T_i^{(c)})$ is evaluated by solving the LPs (18), (17), the derivatives are obtained using results from parametric sensitivity analysis. In particular, we have that

$$\frac{\mathrm{d}V_i(T_i)}{\mathrm{d}t_i^{\mathrm{in}}} = \frac{\partial \mathcal{L}_i(w_i(T_i), \lambda_i(T_i), \nu_i(T_i))}{\partial t_i^{\mathrm{in}}} = \nu_i^{\mathrm{in}}(T_i)\frac{\partial p_i^d\left(t_i^{\mathrm{in}}, w_i(T_i)\right)}{\partial t_i^{\mathrm{in}}}, \tag{29}$$

where $\mathcal{L}_i(w_i, \lambda_i, \nu_i)$ is the Lagrangian Function of the QP (11).[33] Here, $w_i(T_i)$ is the primal solution of (11) for T_i, $\lambda_i(T_i)$ is the dual solution corresponding to the constraints (11b)-(11d), and $\nu_i(T_i) = (\nu_i^{\mathrm{in}}(T_i), \nu_i^{\mathrm{out}}(T_i))$ is the dual solution corresponding to constraints (11e), (11f). The second-order derivatives can then be obtained using the chain rule, for example,

$$\frac{\mathrm{d}^2V_i(T_i)}{\mathrm{d}{t_i^{\mathrm{in}}}^2} = \frac{\mathrm{d}\nu_i^{\mathrm{in}}}{\mathrm{d}t_i^{\mathrm{in}}}\frac{\partial p_i^d\left(t_i^{\mathrm{in}}, w_i(T_i)\right)}{\partial t_i^{\mathrm{in}}} + \nu_i^{\mathrm{in}}(T_i)\left(\frac{\partial^2 p_i^d\left(t_i^{\mathrm{in}}, w_i(T_i)\right)}{\partial {t_i^{\mathrm{in}}}^2} + \frac{\partial^2 p_i^d\left(t_i^{\mathrm{in}}, w_i(T_i)\right)}{\mathrm{d}t_i^{\mathrm{in}}\partial w_i}\frac{\mathrm{d}w_i}{\mathrm{d}t_i^{\mathrm{in}}}\right). \tag{30}$$

The derivatives of the constraints are obtained similarly, as is exemplified for $U(t_i^{\mathrm{in}})$ below. We have by definition that

$$p_i^d(U(t_i^{\mathrm{in}}), w_i^{\mathrm{ub}}(t_i^{\mathrm{in}})) - p_i^{\mathrm{out}} = 0, \tag{31}$$

where $w_i^{\mathrm{ub}}(t_i^{\mathrm{in}})$ is the solution to (17) for t_i^{in}. Differentiation w.r.t. t_i^{in} then gives that

$$\frac{\mathrm{d}U(t_i^{\mathrm{in}})}{\mathrm{d}t_i^{\mathrm{in}}} = -\left(\frac{\partial p_i^d\left(t_i^{\mathrm{in}}, w_i^{\mathrm{ub}}(t_i^{\mathrm{in}})\right)}{\partial t_i^{\mathrm{in}}}\right)^{-1}\frac{\partial p_i^d\left(t_i^{\mathrm{in}}, w_i^{\mathrm{ub}}(t_i^{\mathrm{in}})\right)}{\partial w_i^{\mathrm{ub}}}\frac{\mathrm{d}w_i^{\mathrm{ub}}}{\mathrm{d}t_i^{\mathrm{in}}}. \tag{32}$$

The second-order derivative is obtained by applying the chain rule to (32), but the resulting expression is rather large and is omitted here for brevity, and the interested reader can find it in Reference 26. It should be noted that it includes the term $\mathrm{d}^2w_i/\mathrm{d}{t_i^{\mathrm{in}}}^2$. The computation of the first and second-order derivatives of $L(t^{\mathrm{in}})$ is identical.

3.1.4 | Parametric sensitivity analysis

The expressions (29), (30), and (32) rely on the first and second-order sensitivity of the primal-dual solution with respect to T_i, which acts as a problem parameter in the QP (11) and LPs (18),(17). Note that for the general parametric optimization problem with free variable x and parameter p

$$\min_x q(x) \quad \text{s.t} \quad a(x, p) \geq 0 \tag{33}$$

the KKT conditions are satisfied at the solution, given that some constraint qualification hold. Denoting the primal-dual solution to (33) $z^* = (x^*, \lambda^*)$, the KKT conditions are

$$\nabla\mathcal{L}(z^*, p) = 0, \quad a(x^*, p) \geq 0, \quad \lambda^* \geq 0, \quad a(x^*, p) * \lambda^* = 0, \tag{34}$$

where $\mathcal{L}(z) = q(x,p) - \lambda^\top a(x,p)$ and $*$ denotes element-wise multiplication. In particular, denoting the set of active constraints at the solution $a_{\mathbb{A}}(x^*,p)$ and the corresponding multipliers $\lambda^*_{\mathbb{A}}$, we have that

$$r = \begin{bmatrix} \nabla\mathcal{L}(z^*,p) \\ a_{\mathbb{A}}(x^*,p) \\ a_{\mathbb{A}}(x^*,p) * \lambda^*_{\mathbb{A}} \end{bmatrix} = 0, \tag{35}$$

and note that the solution map $z^* := z(p)$ must be such that $\frac{\mathrm{d}r(z(p),p)}{\mathrm{d}p} = 0$. Evaluating the total derivative of $r(z(p),p)$ w.r.t. p, one obtains

$$\frac{\mathrm{d}z}{\mathrm{d}p} = -\left(\frac{\partial r}{\partial z}\right)^{-1}\frac{\partial r}{\partial p}, \tag{36}$$

and, using the chain rule,

$$\frac{\mathrm{d}^2 z}{\mathrm{d}p^2} = -\left(\frac{\partial r}{\partial z}\right)^{-1}\left(\frac{\mathrm{d}}{\mathrm{d}p}\left(\frac{\partial r}{\partial p}\right) + \frac{\mathrm{d}}{\mathrm{d}p}\left(\frac{\partial r}{\partial z}\right)\frac{\mathrm{d}z}{\mathrm{d}p}\right). \tag{37}$$

Both (36) and (37) exist if $\left(\frac{\partial r}{\partial z}\right)^{-1}$ exists, and the latter is guaranteed if linear independence constraint qualification and the second-order sufficient conditions hold at the primal-dual solution of (33). The sensitivities required to evaluate (29), (30), and (32) can thereby be obtained by solving (36) for the KKT conditions of the QP (11) using the parameters T_i, and by solving (36) and (37) for the KKT conditions of the LPs (17), (18), using the parameter t_i^{in}.

Note that if the program (33) is solved using a second-order method, the solver performs iterations similar to

$$z^{(c+1)} = z^{(c)} - \alpha^{(c)}\left(\frac{\partial r}{\partial z}\right)^{-1} r. \tag{38}$$

This means that the solver will have factorized the matrix $\frac{\partial r}{\partial z}$ at the solution z, whereby the evaluation of (36) can be performed at little additional computational cost. Consequently, if a second-order method is used to evaluate $V_i(T_i)$ and $h_i(T_i)$ in the SQP, the derivatives are very cheap to compute.

3.2 | Schematic algorithm

In a practical setting, the SQP procedure is performed as follows

1. In the central node, Initialize the primal-dual variables $z^{(0)} = (\boldsymbol{T}^{(0)}, \mu^{(0)})$.
2. $\forall\, i$: On each vehicle independently, solve QP (11) and LPs (18), (17) to get cost, constraints and derivatives.
3. In the central node, assemble and solve QP (23) to get $\Delta z^{(c)}$
4. Perform Projection Line search according to Section 3.1.2:
   a. $\forall\, i$: On each vehicle independently, solve LPs (18), (17) at $T_i^\alpha = T_i^{(c)} + \alpha^{(c)}\Delta T_i^{(c)}$ and compute $\mathcal{P}_i(T_i^\alpha)$. Solve QP (11) with $\mathcal{P}_i(T_i^\alpha)$ to get V_i, ∇V_i, $\nabla^2 V_i$, h_i, ∇h_i, $\nabla^2 h_i$.
   b. In the central node, assemble $M^\alpha = \sum_{i=1}^{N_a} V_i(\mathcal{P}_i(T_i^\alpha) + \rho^{(c)}\|h_s(\mathcal{P}(\boldsymbol{T}^\alpha))\|_1$
   c. In the central node, check if condition (26) is fulfilled. If not, set $\alpha^{(c)} = \alpha^{(c)}\beta$, $\beta \in (0,1)$ and repeat from 44a.
5. In the central node, perform primal-dual update (28a). If a solution is not reached, increment c and repeat from 3.

The computations in steps 2 and 44a consist of the solution of QP (11) and LPs (17),(18) and the associated derivative computations detailed in Section 3.1.3. We emphasize that most computations are separable and can be performed in parallel on board the vehicles. However, while the LPs (17)-(18) can be solved in parallel they must be solved before the QP (11) due to the use of $\mathcal{P}_i(T_i)$. The nonparallelizable part of the algorithm is the formation and solution of the QP-subproblem (23), which thereby necessitates a central network node. In the scenarios considered in this article, the

Data	$\mathcal{P}(T_i)$	V_i	$\nabla_{T_i} V_i$	$\nabla^2_{T_i} V_i$	$h_i(T_i)$	$\nabla_{T_i} h_i(T_i)$	$\nabla_{T_i} h_i(T_i)$
# Floats	2	1	2	3	2	2	2

Note: The central node needs to send two floats to each vehicle: the current primal solution candidate $T_i^{(c)}$.

TABLE 1 The information which needs to be sent from a vehicle to the central node each iterate

Coordinator shown in Figure 1A takes this role. Note that the SQP sub problem (23) has $2N_a$ variables and $5N_a - 1$ constraints, and will be significantly smaller than the vehicle-level QP (11) in moderately sized scenarios and horizons N. In such cases, the computational bottleneck will be the evaluation of $V_i(T_i^{(c)})$ and $h_i(t_i^{\text{in}\,(c)})$ and the associated derivatives.

A convergence proof for the SQP applied to (7) is given in Reference 27.

3.2.1 | Communication aspects

The information that needs to be communicated from the central node to the vehicles is only the currently held primal solution candidate, consisting of $2N_a$ floats. The data a vehicle is required to communicate to the central node is listed in Table 1. Consequently, each iterate will involve the communication of at least 14 floats per vehicle, and additional 14 for each reduction of the step size α. This will increase the time per iterate, and can, depending on communication protocol and implementation, constitute the bulk of the time required to solve the problem.

We note that it is only necessary to resend $\mathcal{P}(T_i^\alpha)$ and $V_i(\mathcal{P}(T_i^\alpha))$ to evaluate the merit function M^α and check the Armijo condition (26). The remaining information in Table 1 could then be sent after the primal-dual update. However, while the amount of data transmitted would be less, two additional rounds of communication would be needed: one from the central node, notifying the vehicles of step acceptance, and one from the vehicles containing the remaining information. As practical communication systems include a significant overhead, the total time required by communication would be notably higher. While this requires the vehicles to compute the derivatives of $V_i(T_i)$ and $h_i(T_i)$ when not strictly needed, the calculations can be made highly efficient and will have a small impact on the total solution time (c.f. Section 3.1.3).

Besides what is mentioned here, the algorithm will require the communication of a number of logical variables, for example, commands for algorithm start and stop, step acceptance, and so on. However, the amount of such data per iterate consists of a few bytes and, since it can be sent together with the other data, the additional time required is negligible.

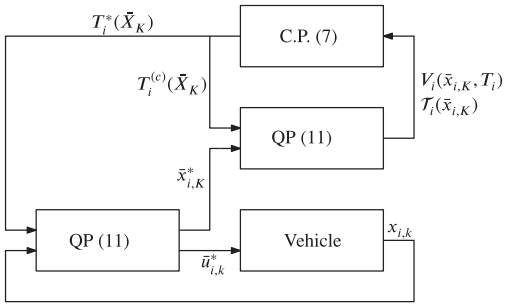
3.3 | Real-time implementation of the intersection-level controller

When the intersection-level control loop is closed using the SQP introduced in this section, it includes distributed computation and wireless communication. In the ideal case, algorithmic overhead, computation and communication require a negligible amount of time compared with the time scale of the system dynamics, and the current vehicle state $x_{i,k}$ (and therefore $V_i(x_{i,k}, T_i)$ and $h_i(x_{i,k}, T_i)$ can be considered constant during the time it takes to solve the SQP. However, non-negligible delays can be expected in a real application, and the vehicle state might change significantly between the SQP iterates. The use of wireless communication in particular is a source of comparatively large delays, as packet drops are likely to occur and subsequent retransmissions of data often are necessary.

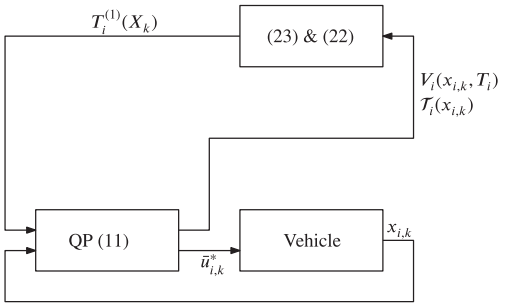
This raises what is known as the real-time dilemma:[34] Should the NLP be solved to convergence when the resulting $\boldsymbol{T}$ will be outdated w.r.t. the system state, or should an approximate solution $\boldsymbol{T}$ be sought using the most up-to-date information? This consideration implies that the resulting control law $\boldsymbol{T}^*(x_{i,k})$ will be a suboptimal approximation of the truly optimal solution feedback, regardless of how the dilemma is handled. In this article, we consider two different solutions to the dilemma, which we present in the following.

3.3.1 | Alternative 1: Solving the intersection-level NLP to convergence from a predicted state

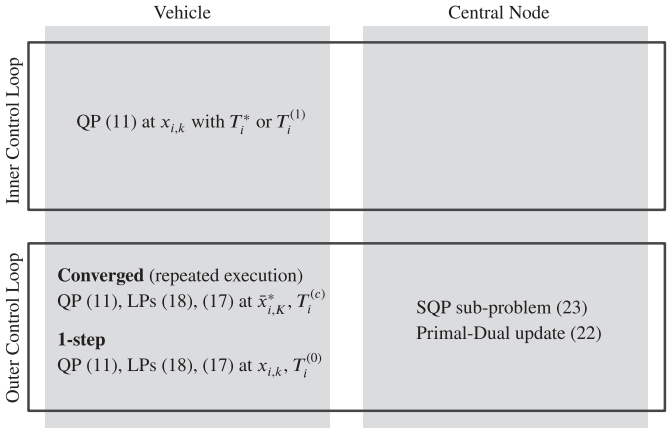
In the first solution, which we denote the *Converged* controller, we use a scheme similar to that of Reference 35 or 36, where $V_i(T_i)$, $h_i(T_i)$ are computed from a predicted future state $\overline{x}_{i,K}$ rather than the current state $x_{i,k}$, $K > k$, and the resulting control law $\boldsymbol{T}^*(\overline{X}_{i,k})$ is not applied until time t_K. The predicted state is obtained from the open-loop predictions of QP (11) in the vehicle-level control loops, which are computed using a previous time-slot schedule $\boldsymbol{T}$. A block diagram of the scheme is shown in Figure 2A.



(A) Schematic illustration of the converged controller. Due to non-negligible delays caused by computation, communication and overhead, the QP (11) solved at the current state cannot be used to provide the sensitivities. The sensitivities are instead computed by solving (11) from the predicted state at future time K.



(B) Structure of the 1-step controller where (7) isn't solved to convergence. The derivatives needed to solve the QP sub-problems (23) are computed from the solution to the QP (11), which also gives the control command.



(C) Illustration of locality of computations in the two intersection-level controllers. Note that in the 1-step controller, the sensitivities required to solve the SQP sub-problem (23) can be obtained from the QP (11) solved to close the vehicle-level control loop.

FIGURE 2 Schematic illustration of the two intersection-level controllers and locality of computation

If the evolution of the real system stays close to the predicted trajectories, that is, $\bar{x}_{i,K}$ predicted at some $k < K$ is close to the actual state $x_{i,K}$ at K, the resulting intersection-level control-law will provide a good approximation $\boldsymbol{T}^*(\overline{X}_{i,k})$ to $\boldsymbol{T}^*(X_{i,k})$. However, the scheme will introduce a significant delay in the feedback of the intersection-level scheme. In fact, denoting the update frequency of the intersection-level controller t_s^O, we note that the reaction to a perturbation that occurs between kt_s^O and $(k+1)t_s^O$ will not be applied to the system until $(k+2)t_s^O$.

3.3.2 | Alternative 2: Approximate the intersection-level NLP in an RTI-fashion

The second solution, which we denote the *1-step* controller, does not solve the SQP to convergence, and thereby avoids long solution times. Instead, we adopt a strategy where the current state measurements are used to compute $V_i(T_i)$, $h_i(T_i)$ and their derivatives, but where only one full SQP step is taken in the solution of (7). The resulting control law $\boldsymbol{T}^{(1)}(X_k) = \boldsymbol{T}^{(0)} + \Delta\boldsymbol{T}^{(0)}(X_k)$ is thereafter applied directly to the vehicles. While the control law is approximate, it enables rapid feedback and reaction to perturbations to the vehicles. The procedure is shown in Figure 2B.

This can be seen as an application of the RTI scheme for dynamic optimization,[30] applied to the NLP (7). However, we note that the dynamic optimization problems (11) that compute the vehicle control commands are solved to convergence at all times.

The physical location of the solution of the optimization problems involved in the inner, vehicle-level control loop as well as the two variations of the outer, intersection-level control loop is provided in Figure 2C.

4 | EXPERIMENTAL VALIDATION

In this section, we describe an experimental setup which was used to validate the bi-level controller described in Section 2.3. In particular, we detail an implementation of the semi-distributed SQP described in Section 3.1 in which most computations are performed on board the vehicles and communicated to a central coordinating unit using V2V communication. We also provide details on the hardware platform used.

4.1 | Practical implementation of the vehicle-level control-loop

The vehicle-level control loop consists of solving problem (11) every time instant based on the current state $x_{i,k}$, using the time-slot T_i and applying the resulting optimal $u_{i,k}$ to the vehicle. However, problem (11) differs from standard MPC formulations in that the position constraints (11e) and (11f) force the vehicle to be at a specific position at a given time. As the vehicle gets closer to the intersection, the ability to affect when the intersection is entered and departed diminishes. Moreover, in a real scenario, the closed-loop system is constantly exposed to perturbations in the form of plant-prediction model mismatches, measurement noise and other external disturbances. It is therefore increasingly likely that problem (11) is infeasible for $(x_{i,k}, T_i)$ as the vehicle gets closer to the intersection. To address this issue we first relax the equality constraints (11e) and (11f) to the inequalities

$$p_i^{\text{in}} - p_i^d(t_i^{\text{in}}, w_i) \geq 0, \quad p_i^d(t_i^{\text{out}}, w_i) - p_i^{\text{out}} \geq 0. \tag{39}$$

With this relaxation, the vehicle is allowed to occupy the intersection within T_i rather than using the intersection throughout all of T_i. While this ensures that the controller, for example, does not slow down the vehicle to stay longer in the intersection in response to a perturbation, infeasibilities are still possible. We therefore introduce a softening of the constraints (39) as

$$p_i^{\text{in}} - p_i^d(t_i^{\text{in}}, w_i) + \sigma_i^{\text{in}} \geq 0, \quad p_i^d(t_i^{\text{out}}, w_i) - p_i^{\text{out}} + \sigma_i^{\text{out}} \geq 0, \quad \sigma_i^{\text{in}} \geq 0 \quad \text{and} \quad \sigma_i^{\text{out}} \geq 0, \tag{40}$$

and add the term $P_i(\sigma_i) = \frac{1}{2}\sigma_i^\top \sigma_i \phi_i^q + \phi_i \mathbf{1}^\top \sigma_i$ to the objective, where $\sigma_i = (\sigma_i^{\text{in}}, \sigma_i^{\text{out}})$ and $\phi_i^q > 0$, $\phi_i > 0$ are penalty weights.

The relaxed and softened vehicle MPC problem solved at time t_k is thereby

$$\begin{aligned}
\underset{\overline{w}_i, \sigma_i}{\min} \quad & J_i(\overline{w}_i) + P_i(\sigma_i) && \text{(41a)}\\
\text{s.t.} \quad & \bar{x}_{i,k} = x_{i,k} && \text{(41b)}\\
& \bar{x}_{i,k+n+1} = A_i \bar{x}_{i,k+n} + B_i \bar{u}_{i,k+n}, \quad n \in \mathbb{I}_{[0,N-1]}, && \text{(41c)}\\
& D_i \bar{x}_{i,k+n} + E_i \bar{u}_{i,k+n}(t) \geq b_i, \quad n \in \mathbb{I}_{[0,N-1]}, && \text{(41d)}\\
& p_i^{\text{in}} - p_i^d(t_i^{\text{in}}, \overline{w}_i) + \sigma_i^{\text{in}} \geq 0, && \text{(41e)}\\
& p_i^d(t_i^{\text{out}}, \overline{w}_i) - p_i^{\text{out}} + \sigma_i^{\text{out}} \geq 0, && \text{(41f)}\\
& \sigma_i^{\text{in}} \geq 0, \quad \sigma_i^{\text{out}} \geq 0. && \text{(41g)}
\end{aligned}$$

Here, we differentiate the state and control of the vehicle $x_{i,k}$, $u_{i,k}$ from the open-loop predictions $\bar{x}_{i,k}$, $\bar{u}_{i,k}$. The control command applied at time t_k is the optimal open-loop control command $u_{i,k} = \bar{u}_{i,k}^*$.

The softening of the constraints ensures that there will be no feasibility issues due to the position constraints (39). In fact, T_i no longer affects the feasibility of the optimization problem (41). Note that if ϕ_i is chosen large enough, $P_i(\sigma_i)^\dagger$ is a so-called exact penalty function.[32] A well-known property of exact penalty functions is that the problem with softened constraints will return a solution with $\sigma_i = 0$ whenever such a solution exists (see eg, 37, Theorem 14.3.1). Moreover, when ϕ_i is chosen large enough and no solution exists where $\sigma_i = 0$, the solution minimizes $\|\sigma_i\|_\infty$ and thereby the violations of

†The quadratic term in $P_i(\sigma_i)$ is added for numerical reasons, and the parameter ϕ^{q} is typically small.

the constraint (39).[25] The quadratic term with parameter ϕ_i^q does not jeopardize the exact penalty property and is added to ensure the numerical stability of the algorithm.

4.2 | Efficient solution of the QPs and LPs

General purpose solvers are too slow to give real-time feasible solutions to the vehicle-level QP (41) and the LPs (18), (17), and solvers tailored to the special structure of these optimization problems have to be considered. As discussed in Section 3.1.3, in case second-order optimization methods are used to solve these QPs/LPs, the sensitivities of the cost function and constraints can be easily and cheaply computed from the local optimization problems by reusing the KKT matrix factorization available from the QPs/LPs solver. In this work, we used a version of the interior-point method (IPM) called HPMPC[29] which is tailored to allow the efficient computation of the tangential predictor at the solution. HPMPC provides an implementation of a Mehrotra's predictor-corrector IPM tailored for the solution of QPs in the form of OCPs. The IPM employs a backward Riccati recursion for the efficient computation of the search direction. As its linear algebra framework, HPMPC makes use of BLASFEO,[38] which provides a set of linear algebra routines tailored to provide high computational performance for rather small matrix sizes, as typical in embedded optimization applications.

On the algorithmic side, the IPM in HPMPC is coupled with a partial condensing algorithm. Partial condensing[39] is a technique that allows one to control the level of sparsity of an OCP problem by trading off horizon length with input vector size, by condensing block-wise the original OCP. It is possible to compute the theoretical optimal horizon length based on the analysis of the flop count of the algorithm. In practice, however, other factors affect the optimal choice of the horizon length, such as the performance of linear algebra routines.[40] The QP (41) is a perfect example of that. Since the state and input vector sizes are very small and the horizon length is long, partial condensing gives a QP reformulation that HPMPC can solve much faster, since many operations on small matrices (where the linear algebra performs poorly) are replaced with few operations on large matrices (where the linear algebra gives higher computational performance).

In this work, HPMPC has been modified to allow the efficient computation of sensitivities. Namely, the solver now allows the reuse of the last KKT matrix factorization (where Lagrange multipliers and slack variables of inequality constraints are fixed at their value close to the solution) to cheaply compute the solution of other systems of linear equations with different right-hand side. If there are no changes in the active set, this allows the efficient computation of the tangential predictor around the current solution.[41] Therefore, the sensitivities in (36) can be cheaply computed by performing the partial condensing of the right-hand side and the solution of the KKT system reusing the cached KKT matrix factorization. The computational cost of these operations is negligible with respect to the QP/LP solution, which comprises a complete partial condensing preprocessing step, plus a KKT matrix factorization and two KKT system solutions per IPM iteration (which are typically in the range of 6-15 per QP/LP solution).

4.2.1 | Efficient solution of the intersection-level problems

As noted in Section 3.2, the computational bottleneck of the SQP algorithm will commonly be the solution of the vehicle-level QP (41) and the LPs (18), (17). This is due to the comparatively small size of the QP subproblem (23) in such cases. General purpose QP solvers can therefore be fast enough and used to solve (23) in real-time. Due to this, MATLABs QP-solver quadprog was used in the experimental validation. However, with an increasing number of vehicles, the time required to solve the QP subproblem (23) with a general purpose solver will approach that required by HPMPC for the solution of (41) and the LPs (18), (17). For large scenarios, solving (23) could therefore become a computational bottleneck of the SQP. It is therefore desirable to use efficient, structure exploiting solvers also for the QP subproblem (23). For this reason, we propose the following reformulation of (23):

$$\min_{\Delta \boldsymbol{T}, u^t} \quad \sum_{i=1}^{N_a} \frac{1}{2} \Delta T_i^\top H^{(c)} \Delta T_i + \nabla_{T_i} f(T_i^{(c)})^\top \Delta T_i \tag{42a}$$

$$\text{s.t.} \quad t_i^{\text{in}(c)} + \Delta t_{i+1}^{\text{in}} = u_i^t \tag{42b}$$

$$u_i^t - t_i^{\text{out}(c)} - \Delta t_i^{\text{out}} \geq 0 \tag{42c}$$

$$h_i(T_i^{(c)}) + \nabla_{T_i} h(T_i^{(c)})^\top \Delta T_i \geq 0. \tag{42d}$$

That is, the time-slot increments ΔT_i are formulated as states in the dynamical system (42b), where the variable $u^t = (u_1^t, \dots, u_{N_a}^t)$ is introduced as a fictitious control. The precedence constraint (7c) is formulated as the path constraint (42c). The problem is thereby written on a stage-wise form for which efficient solvers such as HPMPC can be deployed, and significant performance gains can be made. For instance, the typical time required to solve (23) for a three vehicle scenario with quadprog is around 2 ms using a standard laptop. The time required by HPMPC for the same problem using the same hardware lies around 40 μs. Moreover, the time-complexity of HPMPC is linear in the number of stages, and approximately 100μs are required for a 30-vehicle scenario, where quadprog requires 7ms. For a 300-vehicle scenario HPMPC requires approximately 850μs to converge, while quadprog requires approximately 1second.

4.3 | Algorithm

The procedures executed by the coordinator and the vehicles when the converged intersection controller is used are summarized in Algorithms 1 and 2. Before the SQP is solved the first time, the central node requests the vehicles' noncoordinated optimal time-slot T_i which serves as the primal initial guess. In subsequent solutions of the SQP, the previous optimal solution is used instead. The solution is considered found when either $\|(\nabla \mathcal{L}(z), h^-(\boldsymbol{T}))\|_\infty < \varepsilon$ or $\|(\Delta \boldsymbol{T}, h^-(\boldsymbol{T}))\|_\infty < \varepsilon$. On the vehicle side, the MPC problem (41), is solved every t_s using the most recently commanded time-slot T_i^*. When a request from the central node is registered, the vehicles evaluate and send the functions and derivatives needed to solve the SQP.

Algorithm 1. Central Node. Here, $\mathcal{D}_i(T_i) = \{V_i(T_i), \nabla_{T_i} V_i(T_i), \nabla^2_{T_i} V_i(T_i), h_i(T_i), \nabla_{T_i} h_i(T_i), \nabla^2_{T_i} h_i(T_i)\}$, where the dependence on the initial state $\hat{x}_{i,0}$ has been dropped for notational simplicity

Send coordination start state x_i^{start} and start time t^{start}.
Request $T_i^{(0)}$ from vehicles, initialize $\mu^{(0)}$, $K = 0$
loop
    Wait until $t == t^{\text{start}} + (K-1)t_s^O$
    Set $t_K = Kt_s^O$, $\boldsymbol{T} = \boldsymbol{T}^*$, $\mu = \mu^*$, $r = 0$
    Broadcast $\boldsymbol{T}, t_K, r$ and request $\mathcal{D}_i(T_i)$ computed at t_K.
    Wait until all vehicles has responded.
    while exit conditions not fulfilled **do**
        Set $r = r + 1$, Assemble and solve (23) for Δz, set $\alpha^{(c)} = 1$
        loop
            Broadcast $T_i^\alpha = T_i^{(c)} + \alpha^{(c)} \Delta T_i^{(c)}$ and request $\mathcal{D}_i(T_i^\alpha)$ and $\mathcal{P}_i(T_i^\alpha)$
            Wait until all vehicles have responded.
            $M^\alpha = V(\mathcal{P}(\boldsymbol{T}^\alpha)) + \rho^{(c)} \|h_s(\mathcal{P}(\boldsymbol{T}^\alpha))\|_1$
            if $M^\alpha \leq M(\boldsymbol{T}^{(c)}) + \gamma D_{\Delta \boldsymbol{T}^{(c)}} M(\boldsymbol{T}^{(c)}) \alpha^{(c)}$ **then**
                $\boldsymbol{T}^{(c+1)} = \mathcal{P}(\boldsymbol{T}^\alpha)$
                $\mu^{(c+1)} = \mu^{(c)} + \alpha^{(c)}(\tilde{\mu}^{(c)} - \mu^{(c)})$
                Exit loop.
            else
                $\alpha^{(c)} = \alpha^{(c)} \beta$
            end if
        end loop
    end while
    Set $\boldsymbol{T}^* = \boldsymbol{T}^{(c)}$ and send out time-slot to apply $\boldsymbol{T}^*$.
    $K \leftarrow K + 1$
end loop

Algorithm 2. Vehicle

1: **loop**
2: Estimate current state $x_{i,0}$, Get synchronized time t
3: **if** Central node sends new time-slot to apply **then**
4: Receive T_i^*
5: $T_i^{\text{local}} = T_i^* - t$
6: **end if**
7: Solve (41) with $(x_{i,0}, T_i^{\text{local}})$
8: Apply optimal $u_{i,0}$ to vehicle
9: **if** Central node request $\mathcal{D}_i(T_i)$ **then**
10: Receive T_i, t_K, r
11: $\tilde{T}_i^{\text{local}} = T_i - t_K$
12: **if** r == 0 **then**
13: Store $x_{i,K}$ from prediction at $t_K - t$ computed on Line˜7
14: Solve LPs for $L^{\text{in}}(x_{i,K})$ and $U^{\text{in}}(x_{i,K})$
15: **end if**
16: Solve LPs (18), (17) and evaluate $h_i(\tilde{T}_i^{\text{local}})$, $\nabla_{T_i} h_i(\tilde{T}_i^{\text{local}})$, $\nabla^2_{T_i} h_i(\tilde{T}_i^{\text{local}})$.
17: Compute $\mathcal{P}_i(\tilde{T}_i^{\text{local}})$ through (27).
18: Solve QP (11) with $(x_{i,K}, \mathcal{P}_i(\tilde{T}_i^{\text{local}}))$ and compute $V(x_{i,K}, \mathcal{P}_i(\tilde{T}_i^{\text{local}}))$, $\nabla_{T_i} V(x_{i,K}, \mathcal{P}_i(\tilde{T}_i^{\text{local}}))$ and $\nabla^2_{T_i} V(x_{i,K}, \mathcal{P}_i(\tilde{T}_i^{\text{local}}))$.
19: Send $\mathcal{P}_i(\tilde{T}_i^{\text{local}})$ and $\mathcal{D}_i(\mathcal{P}_i(\tilde{T}_i^{\text{local}}))$ to central node.
20: **end if**
21: **end loop**

4.3.1 | Implementation restrictions and practical considerations

Due to implementation related details, the algorithm was executed at a sampling time of $t_s = 0.1$ s. This restricted the communication of new information from both the central node and the vehicles to occur at a maximum of 10 Hz. With an ideal communication system, one iteration of the SQP (with $\alpha^{(c)} = 1$) therefore requires t_ss to broadcast $\boldsymbol{T}$ (line 11 of Algorithm 1) and t_s s for the vehicles to respond with $\mathcal{D}_i(\mathcal{P}_i(T_i^\alpha))$ (line 12 of Algorithm 1), that is, the lowest time required per SQP iterate is $2t_s$ s. Moreover, for the converged intersection controller, each execution of the SQP commences t_s^O s before the resulting time-slots should be applied to the vehicles, which gives the algorithm t_s^O s to converge and notify the vehicles of the results. For simplicity, the period of the intersection-level control loop is set to t_s^O s, which, due to the long expected solve times, is set to $t_s^O = 3$ s. Finally, to ensure that all vehicles are predicted to be before the intersection when the new time-slots are applied, the SQP is only solved when no vehicle is close to the intersection. In particular, for scenarios where the desired speed is $v^{\text{ref}} = 50km/h$, the SQP is suspended when the first vehicle is 50 m away from the intersection. We emphasize that this modification is done for simplicity of implementation and that the problem formulation allows the SQP to be solved with a vehicle to be inside the intersection.

4.4 | Test platform

The coordination controller was tested at the Asta Zero proving ground outside Gothenburg, Sweden. The test platform consisted of the three different Volvo vehicles shown in Figure 3A: One Volvo S60 T5 Petrol Turbo sedan, one Volvo S60 D5 Turbo Diesel sedan and one Volvo XC90 T6 Petrol Turbo SUV. All cars were equipped with automatic gearboxes and an interface for external control of the longitudinal motion. In particular, all vehicles were commanded by supplying a desired longitudinal acceleration to a controller, which thereafter sent the appropriate commands to the engine, gear-box, and friction brakes. The vehicles had an on-board sensor suite consisting of wheel encoders, inertial measurement units and real-time kinematic GPS receivers. The latter was capable of providing positioning estimates with measurement error

(A) (B)

FIGURE 3 Photos of the experimental hardware platform. A, Shows the three vehicles used, B, shows the hardware installation in one of the vehicles. The 3G router was used to provide an IP link for the real-time kinematic corrections used by the GPS receiver [Colour figure can be viewed at wileyonlinelibrary.com]

standard deviations as low as $\sigma_{\text{GPS}} = 0.05$ m as well as global-time synchronization. To improve the positioning estimates and handle sporadic GPS outages, Extend Kalman Filters based on that presented in Reference 42 were used in all vehicles to fuse the available sensor data. The one-dimensional position $p_{i,k}$ was constructed by first projecting the estimate of the global-position onto a reference path along the road and then taking $p_{i,k}$ as the geodesic distance along the path from the projected point to the start of the intersection.

Moreover, each vehicle was equipped with ITS G5 compliant V2V communication equipment from RENDITS.[43] On each vehicle, the experiment software ran on two computational units that communicated with each other using UDP over Ethernet: one MicroAutoBox II (MABx) real-time prototyping platform, interfacing with the vehicle, sensors, and communication equipment which ran the algorithm logic and state estimation, and one MacBook Pro with an Intel i7 4770HQ CPU and 16 GB of RAM, on which the QPs (11), (41) and LPs (17), (18) were solved using HPMPC. The hardware setup in one of the vehicles is shown in Figure 3B.

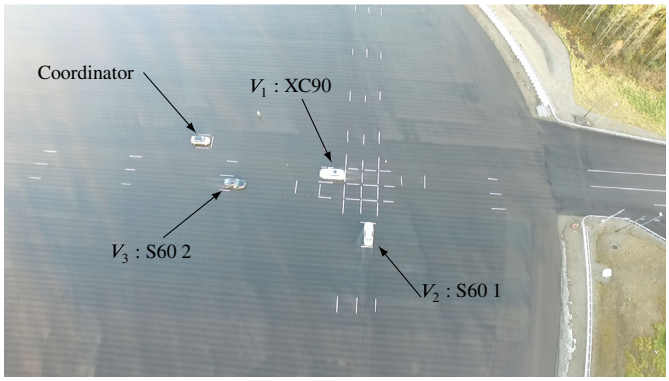
Finally, the experimental setup included a central coordinating unit, consisting of a MacBook Pro laptop with an Intel i5 4308U CPU and 8 GB of RAM, together with a V2V communication device from RENDITS. The central coordinating unit was used to control and monitor the experiments and executed the central parts of the SQP. The latter included solution of the QP-subproblem (23) with quadprog in MATLAB/Simulink 2016b.

4.4.1 | Prediction model, objective, and parameters

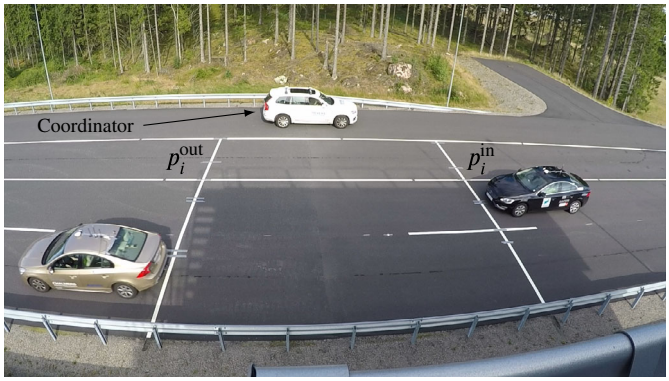
The prediction model used during the experiments was a simple double integrator, $\dot{p}_i(t) = v_i(t)$, $\dot{v}_i(t) = u_i(t)$, where the acceleration is the input and $x_i(t) = (p_i(t), v_i(t))$. For this dynamical system $p_i^d(t, w_i) = p_{i,k} + (t - kt_s)v_{i,k} + \frac{1}{2}(t - kt_s)^2 u_{i,k}$, with $k = \text{floor}(t/t_s)$. Furthermore, we employed the objective function

$$J_i^d(w_i) = (v_{i,N} - v_i^{\text{ref}})^2 Q_i^{\text{f}} + \sum_{i=0}^{N-1} (v_{i,k} - v_i^{\text{ref}})^2 Q_i + R_i u_{i,k}^2, \tag{43}$$

where the desired speed v_i^{ref}, and the objective function weights $Q_i > 0$ and $R_i > 0$ were varied between different instances of the experiment. The state and control were constrained to $v_{i,k} \geq 0$ and $u_{i,k} \in [-4, 1.6] m/s^2$, where the latter was due to limitations in the vehicle actuation interfaces. The vehicle-level control loops were closed with $t_s = 0.1$ s and the horizon length was set to $N = 200$. The objective and prediction model were chosen due to their simplicity. In particular, the dynamics do not include any parameters to identify, and the objective enables an intuitive understanding of how the solution will change with variations to the penalty weights. However, we want to emphasize that these choices are not restrictive and that other linear-quadratic models are possible. For instance, in Reference 22 a prediction model based on a linearization of a nonlinear vehicle model is used together with a quadratic approximation of an economic objective function obtained through the method presented in Reference 44.



(A) Aerial photo of the crossing configuration used in the experimental validation. The white lines before the square representing the intersection illustrates the different safety margins employed.



(B) Photo of the parallel configuration used in the experimental validation. The white lines mark the beginning and end of the intersection, p_i^{in} and p_i^{out}, and collisions are thereby avoided when only one vehicle is between the two white lines at a given time.

FIGURE 4 Photos of the two different configurations used in the experimental validation [Colour figure can be viewed at wileyonlinelibrary.com]

5 | RESULTS

In this section, we present and discuss the results from the experimental campaign, which demonstrate the performance of the semi-distributed SQP and both the Converged and 1-step intersection-level controllers.

In total, more than 80 experiments were performed, where the initial conditions, objective function weights, and other parameters were varied. In all experiment instances, the vehicles were first controlled to a predefined starting state, typically one where a collision would occur if no action was taken, before the bi-level controller was initialized. The experiments were performed in two different modes: in an actual intersection, as shown in Figure 4A and in a parallel configuration where the approaching roads were laid out next to each other and the intersection was represented by a segment on the road, as shown in Figure 4B. The latter was used to enable evaluation of the controller without risk of collision, and is the primary source of the data reported in this section. However, the interested reader can find video material from experiments performed in the crossing configuration at Reference 45.

For comparison, some simulation results of the closed-loop system are also provided in this section. In these cases, the vehicles were simulated with the nominal model.

5.1 | Evaluation of the semi-distributed SQP

In this subsection, we present and analyze data from the implementation of the semi-distributed SQP used in the experiments. We focus on an experimental scenario with the objective function weights $Q_i = 1, R_i = 10$, the reference speed $v_i^{\text{ref}} = 50$ km/h, $\forall i$, and where all vehicles are initiated at $p_{i,0} = -200$ with $v_{i,0} = 50\,km/h$. Data from the first SQP instance, where the problem is solved from $x_{i,0} = (p_{i,0}, v_{i,0}), \forall i$, is shown in Figure 5. In particular, Figure 5A,B show the solutions to the vehicle-level problems corresponding to the iterates $\boldsymbol{T}^{(c)}$ of the SQP. The two horizontal lines in Figure 5A represent the beginning and end of the intersection, that is, collisions are avoided when at most one trajectory is between the lines at all times, and the primal iterates $\boldsymbol{T}^{(c)}$ are shown as vertical lines.

The solver is initialized at the uncoordinated solution, where all vehicles keep the constant velocity v_i^{ref} and therefore occupy the intersection simultaneously. As can be seen in Figure 5A, the time-slots $\boldsymbol{T}$ satisfy the order constraints $h_i(\boldsymbol{T}) \geq 0$ already after the first iterate, whereby a collision free solution is available. The subsequent two iterations retain feasibility and improve the solution. Full steps ($\alpha = 1$) are taken in all iterates.

The algorithm progress measures shown in Figure 5C further illustrate this fact: feasibility is reached after the first iterate ($\|h_s^-(\boldsymbol{T})\|$ drops below the set tolerance). Note the relatively loose convergence threshold $\varepsilon = 10^{-3}$, which is selected in relation to the properties of the physical system. In particular, the GPS provides measurements with a positioning error standard deviation around $\sigma_{\text{GPS}} = 0.05$ m. The standard deviation of the error between the commanded time-slot T_i and

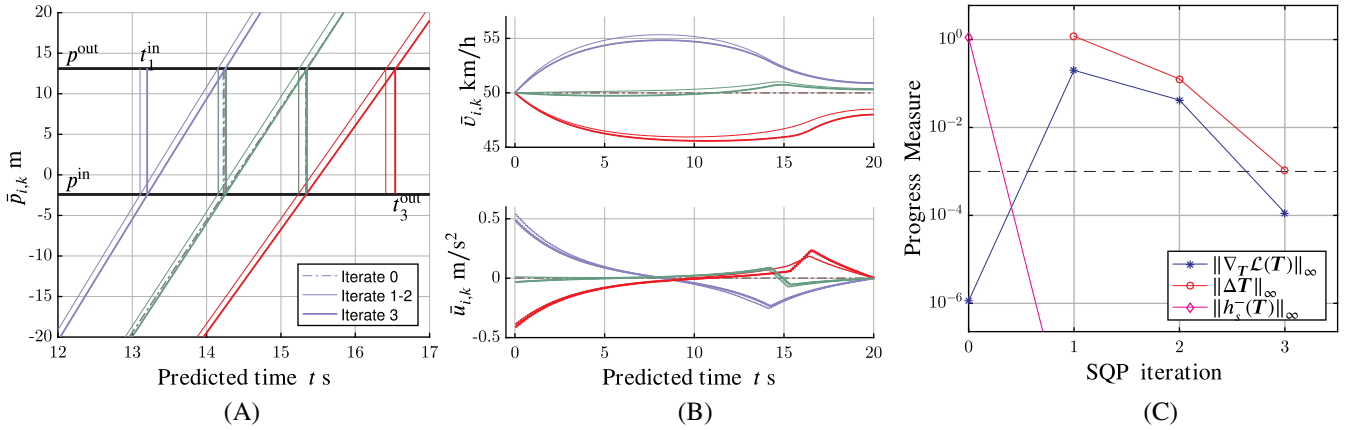


FIGURE 5 Example of the progression of the sequential quadratic programming algorithm in one instance from the experimental campaign. A, Shows the primal iterates $T_i^{(c)}$ as vertical bars and the position trajectory $\bar{p}_{i,k}$ for the solutions to (11) corresponding to the primal iterates, B, shows the corresponding velocity $\bar{v}_{i,k}$ and input $\bar{u}_{i,k}$ sequences, and C, the corresponding algorithm progress measure, where the dashed line is the termination tolerance. The data from the different vehicles in A and B are differentiated by color [Colour figure can be viewed at wileyonlinelibrary.com]

that resulting from the closed-loop application of (11) using T_i will therefore be above one millisecond[‡] for speeds around 50 km/h. Enforcing constraint satisfaction or changes to the primal variables below 10^{-3} $[s]$ will consequently have no noticeable effect on the physical system.

The small impact of small changes in T_i on the vehicle trajectories is further shown in Figure 5A,B, where the solutions of (11) corresponding to iteration 2 and 3 are indistinguishable. In fact, the difference in $\bar{u}_{i,0}$ between iteration 2 and 3 is below the resolution of the actuation interface, that is, the applied control command $\bar{u}_{i,0}$ after iteration 2 and 3 would be interpreted as identical by the vehicle.

5.1.1 | Solution time analysis

The mean computation time was 1.664 ms with $\sigma \approx 1.263$ ms ($n = 52736$) for the LPs, and 1.607 ms with $\sigma \approx 1.107$ ms ($n = 246132$) for the vehicle-level QPs. Since the SQP normally only required a few iterations to converge to a relevant threshold, a solution to (7) should have been found within a few hundredths of a second. However, due to a rudimentary implementation and hardware limitations this was not the case in during the experiments. Instead, the average time required to solve the SQP was 1.740 s with $\sigma \approx 0.406$ s ($n = 130$).

An example of how this time is spent is provided in Figure 6 which shows a timeline from the SQP instance shown in Figure 5. In Figure 6, the width of the bars represents the time spent solving the LPs (18), (17) and the QP (11) for the vehicles, corresponding to Lines 16 and 18 of Algorithm 2, and the time spent solving the QP-subproblem 23 in the central node, corresponding to Line 9 of Algorithm 1. As the figure illustrates, only 55 ms, corresponding to 1.6% of the total time 2.723 s, is spent in computations. The time required for the other operations relevant for the SQP in Algorithms 1 and 2 is negligible, and the remaining 98.4%, that is, the white gaps of Figure 6A, is primarily spent in the waiting states of Lines 7 and 12 in Algorithm 1. Possible explanations for the long delays are inefficient buffer handling in the communication modules of the MABx and packet drops in the wireless links. To a smaller extent, the delays are due to the low communication frequency used and the lack of synchronization between the vehicles and the central node, discussed in Section 4.3.1. The impact of the slow update rate can be seen in a comparison between the computations of vehicle 1 and 2: in iteration 1, occurring around $t = -1.5$ s in Figure 6A, vehicle 1 and 2 perform their computations simultaneously. In iterations 0, 2, and 3, on the other hand, there is delay of $t_s = 0.1$s between the vehicles. The explanation is that a message is processed by Algorithm 2 at time $(k+1)t_s$ or kt_s depending on when it is received relative to the ticks of the local clock. Small variations in the reception time can therefore cause a variation of $t_s = 0.1$ s in the relative time

[‡]This is true for the otherwise ideal case. In reality, other uncertainties are present as well and the performance is in practice therefore even worse.

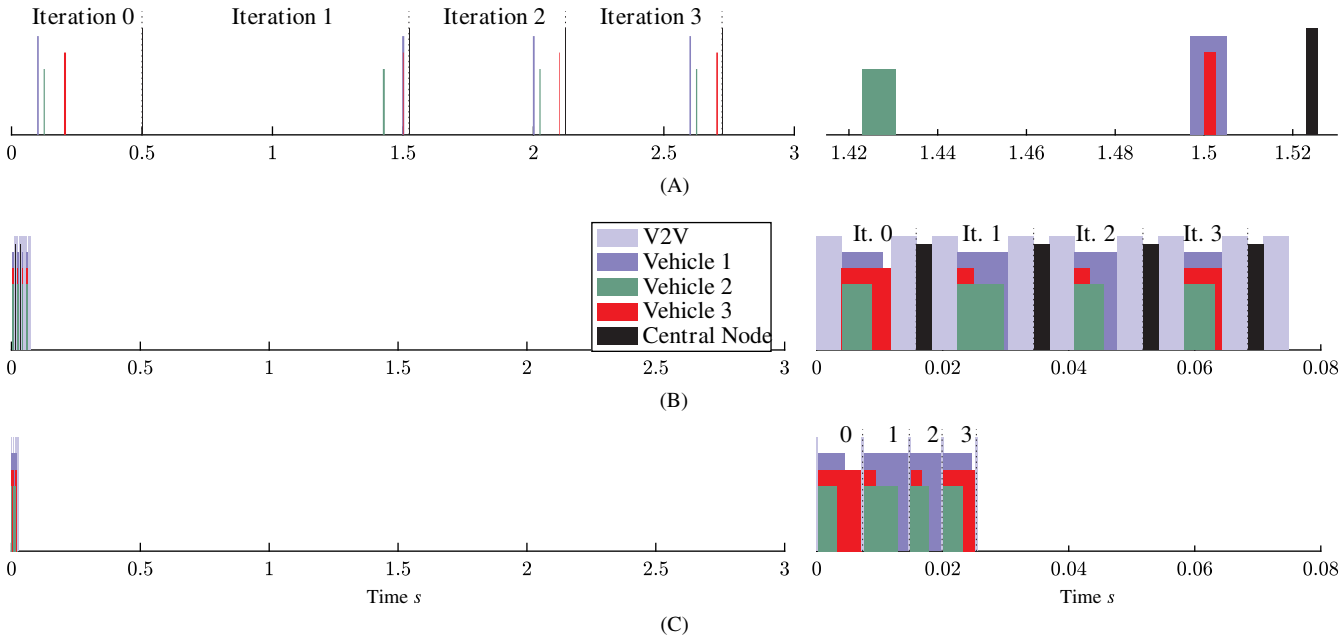


FIGURE 6 Illustration of the time spent on computation in the of the sequential quadratic programming instance shown in of Figure 5, The width of the bars corresponds to the time spent for the different parts of Algorithms 1 and 2. For the vehicles, it consists of the time required to solve (18), (17), and (11) sequentially, and for the central node it consists of the time required to solve (23) with MATLABs quadprog. The bars are shown with different heights to ease visualization. A, shows the timings recorded during the experiment, B, shows the likely timings of an improved implementation, and C, illustrates the timings of an hypothetical ideal implementation which uses a tailored communication protocol [Colour figure can be viewed at wileyonlinelibrary.com]

between the processing of a message in two cars. Moreover, the lack of synchronization can be observed when vehicle 2 and 3 are compared: the computations of vehicle 3 consistently occur around 0.07 s before those of vehicle 2. We want to highlight that the long time required to solve the SQP is almost entirely related to our specific implementation, and performance would improve dramatically with a few modifications of the implementation.

For comparison, we provide an example of the timeline for a more efficient implementation of the SQP in Figure 6B. In this case, the long and unnecessary waits have been removed and the algorithm is synchronized between the vehicles, but everything else is kept unchanged. The time marked as required by communication is taken from Reference 42, where an empirical study is presented on the time performance of the communication system used during the experiment. With such an implementation, the same problem instance would be solved in 0.074 s, where 48% would be spent on communication, 38.5% on the solution of the vehicle-level LPs and QP, and 13.5% on solving the QP-subproblem (23).

Finally, it is reasonable to expect even lower solution times with improvements to both the algorithmic implementation and the equipment. For instance, the use of the reformulation of the QP-subproblem and its solution with HPMPC would, as discussed in 4.2, significantly reduce the time required for the central computations. Furthermore, solving the two LPs (18), (17) in parallel on each vehicle would shorten the time required for the vehicle-side computations. Finally, while the general purpose V2V equipment requires around 4 ms for each transmission, a tailored communication protocol could be made significantly faster. For instance, it is reported in Reference 46 that the time to transmit data using the 802.11 p physical layer could be as low as $t_{\text{com}} = 40 + \text{ceil}\left((n_{\text{data bits}} + 22)/48\right) 8\ \mu s$. Since all vehicles could use different channels and transmit their data to the central node simultaneously, sending 12 floats per vehicle in double precision would thereby take 176 μs. The time-line for a solution of the same SQP instance in this hypothetical setting is shown in Figure 6C, where the solution would be found in 0.0256 s.

5.1.2 | Consistency

The algorithm consistently exhibited the same behavior as in Figure 5, with almost immediate feasibility followed by a few optimality-improving iterations. The algorithm progress measures of a selection of the SQP instances are provided in

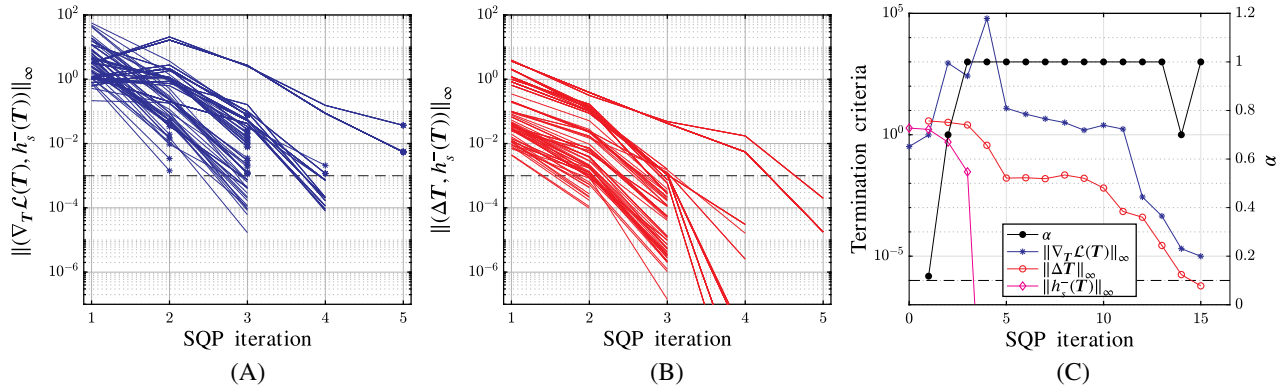


FIGURE 7 A and B, Show the development of the progress measure in a selection of sequential quadratic programming instances from the experimental campaign. In both figures, the stars indicate cases when the algorithm was terminated due to only one progress measure reaching the threshold. C, Shows the progress measure in a case where reduced steps are taken. In A to C, the dashed line is the tolerance $\epsilon = 10^{-3}$ [Colour figure can be viewed at wileyonlinelibrary.com]

Figure 7 for illustration. In particular, the convergence to ϵ was achieved in two iterations in 7% of the instances, in three iterations in 50%, in four iterations in 28%, and in five iterations in 15%.

Finally, we have noticed that reduced steps ($\alpha < 1$) are required only in "hard" scenarios. Examples of "hard" scenarios include those where the coordination is initiated very close to the intersection for the initial vehicle velocity, and those where a large number of vehicles need to cross the intersection simultaneously. For practical reasons we were not able to perform sufficiently hard scenarios during the experimental campaign, due to which full steps were taken in all experimental SQP instances. For illustration purposes, the progress measures of a three-vehicle scenario where reduced steps were taken is given in Figure 7C. This scenario was particularly hard, since $p_{1,0} = -45$ m, $v_{i,0} = 45$ km/h while $p_{2,0} = p_{3,0} = -40$ m, $v_{2,0} = v_{3,0} = 50$ km/h, such that the first and last vehicles were forced to perform very aggressive maneuvers to avoid collision.

5.1.3 | A larger example

To further demonstrate the behavior of the semi-distributed SQP, we present simulated results from a larger problem instance in Figure 8. In this scenario, 12 vehicles are randomly generated at distances between 50 and 200 m from the intersection at 50 km/h. As Figure 8B illustrates, the algorithm exhibits the same behavior as in the smaller scenarios: feasibility, and thereby collision avoidance is reached rapidly, in this case after the second iterate, and thereafter small adjustments toward optimality are performed.

While the implementation of the SQP used in the experiments would require prohibitively long time to solve the problem, it would be solved in 0.224 s with the improved implementation of Figure 6B, and attain feasibility in 0.037 s. Moreover, the problem would be solved in 0.089 s with the ideal implementation discussed in Section 5.1.1, where a feasible solution would be available in 0.015 s.

5.2 | Evaluation and comparison of controllers

In this subsection, we present and analyze both simulated and experimental data on the performance of the bi-level closed-loop controller. In particular, we provide comparisons between the converged and 1-step formulations of the intersection-level controller and study their ability to reject large perturbations.

5.2.1 | Vehicle-level control loop

Regardless of how the time-slots $\boldsymbol{T}$ are computed, the accuracy with which the vehicles conform to the time-slots determines whether or not the closed-loop system is collision free. Even though a simple dynamic model was used in

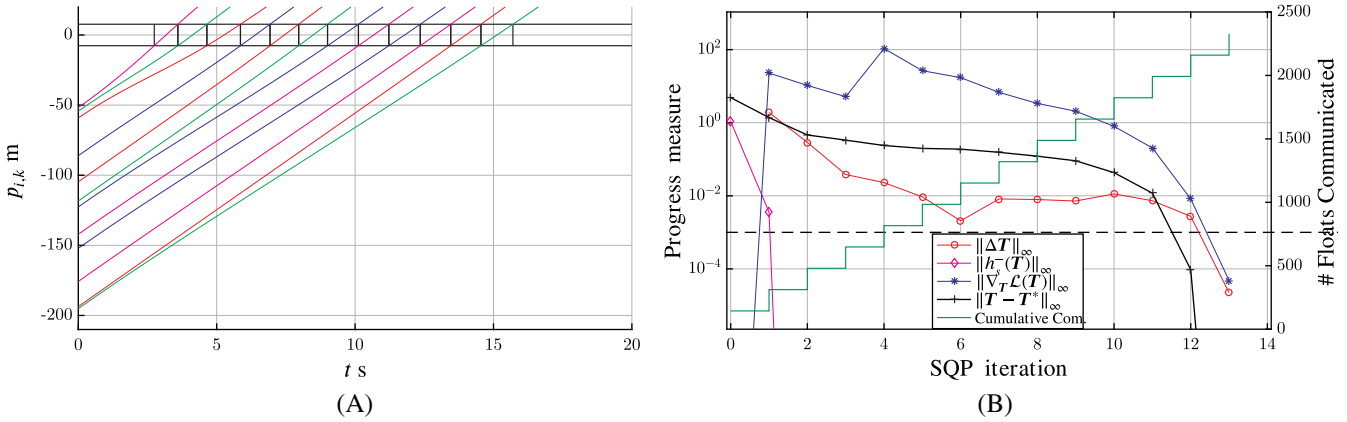


FIGURE 8 Example scenario consisting of 12 cars in a four-way intersection. The positions of the vehicles are shown in A, where the different colors differentiate different lanes. In B, the algorithm performance measures are displayed together with the cumulative number of floating point numbers passed in the two-way communication between the central node and the vehicles [Colour figure can be viewed at wileyonlinelibrary.com]

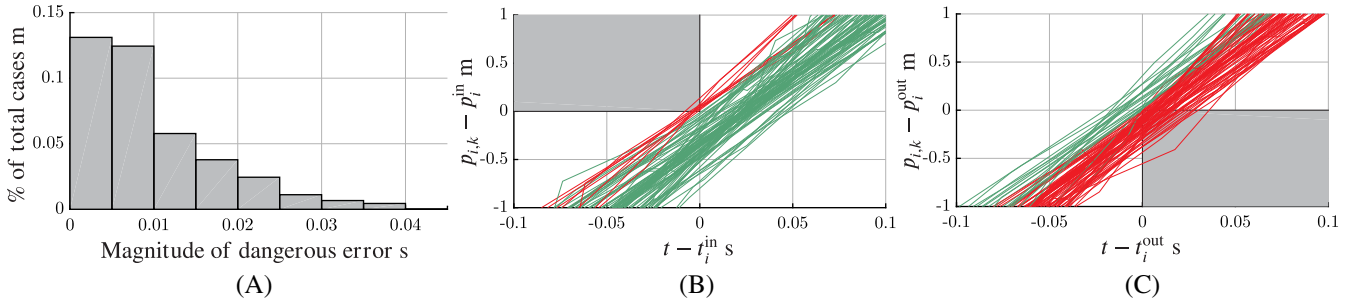


FIGURE 9 Illustration of the vehicle-level model predictive control performance. Statistics on the difference between the commanded and actual t_i^{in} and t_i^{out} is shown in A. In particular, only potentially dangerous violations such that $t_i^{\text{in Actual}} < t_i^{\text{in}}$ or $t_i^{\text{out Actual}} > t_i^{\text{out}}$ are considered. The corresponding position trajectories are shown in B and C. Trajectories that satisfy and violate (41e) and (41e) and (41f) are colored green and red, respectively. The gray area in A and B corresponds to the constraints (41e) and (41f), and are consequently not crossed by safe trajectories. The data is obtained from all three vehicles during 75 experiments and all trajectories are shifted in $p_{i,k}$ and t to ease visualization [Colour figure can be viewed at wileyonlinelibrary.com]

the vehicle-level MPC during the experiment and several nonmodeled nonlinearities were present, the ability of the closed-loop system to satisfy the position constraints (41e) and (41f) was remarkable. In particular, the difference between the commanded (T_i) and actual (T_i^{Actual}) time-slot was small in almost all cases. Combining the evaluation of $t_i^{\text{in Actual}} - t_i^{\text{in}}$ and $t_i^{\text{out Actual}} - t_i^{\text{out}}$ for all three vehicles and all experiment runs, 450 constraint violation data-points were collected. The error was on the potentially dangerous side, that is, vehicles entering the intersection too early ($t_i^{\text{in Actual}} < t_i^{\text{in}}$) or leaving too late ($t_i^{\text{out Actual}} > t_i^{\text{out}}$), in 43.3% of the cases. The distribution of the error is shown in Figure 9A, and we emphasize that for more than 90% of the potentially dangerous constraint violations, the errors were below 0.03 s. To illustrate how this small error translates to collision risks, the corresponding trajectories in the time-position space are given in Figure 9B,C for t_i^{in} and t_i^{out}, respectively.

As shown in Figure 9C, the most dangerous constraint violations occur for the out time constraint (41f), that is, such that the vehicles leaves the intersection too late. Note, however, that the magnitude of violations only are such that a few decimeters of the leaving vehicle remain inside the intersection when the following vehicle enters. That violations are larger for the out time constraints is likely due to the (unmodeled) actuator dynamics being faster in deceleration than in acceleration (c.f. friction brakes and internal combustion engine). Successful compensation of errors due to measurement noise and prediction model inaccuracies can therefore be made closer to the intersection when these cause the vehicle to enter early rather than leave late. It is expected that all constraint violations could be decreased by using a more sophisticated prediction model in (41), more accurate sensors and a higher update frequency t_s. Finally, we note that a

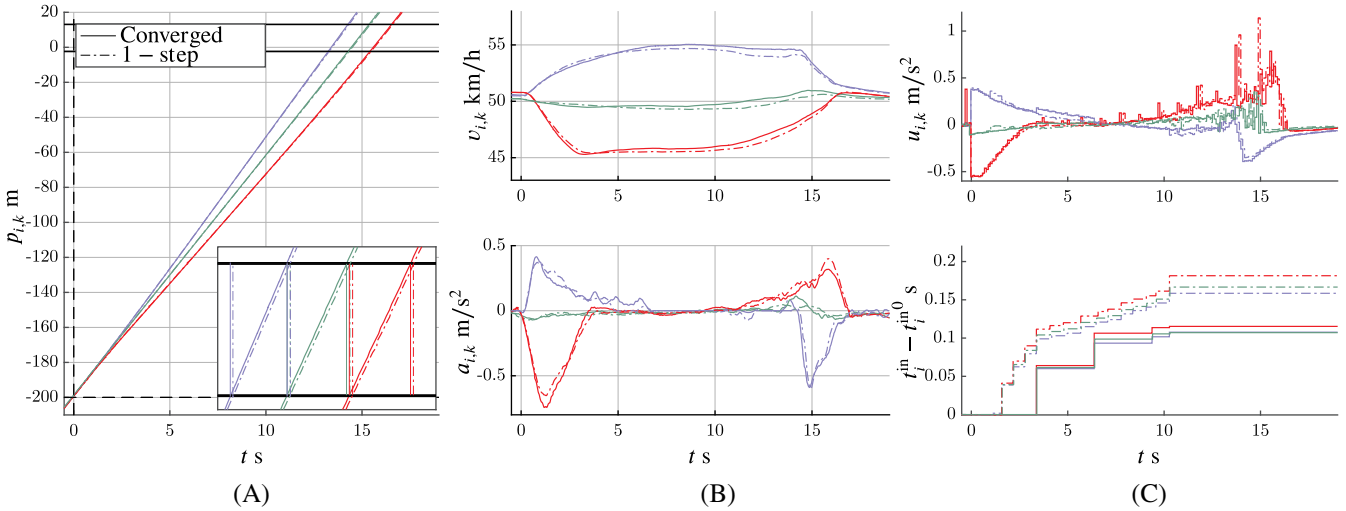


FIGURE 10 Data from two experimental runs using the 1-step and converged intersection-level controllers, where $Q_i = 1$, $R_i = 10$, $v_i^{\text{ref}} = 50$ km/h, $\forall i$. The position trajectories are shown in A and the velocity and acceleration trajectories in B. The upper plot of C shows the output of the vehicle-level MPC controllers, whereas the lower plot illustrates the output of the intersection-level controller. To promote visibility, the lower plot in C shows only changes in t_i^{in} with respect to the initial schedule $t_i^{\text{in}^0}$, which is the same for both controllers. For both controllers, the time-slot schedule is frozen at $t \approx 10$s. Due to substantial noise levels, the acceleration data in B has been smoothened with a noncausal filter to promote visibility. Finally, the noise in the control signals of vehicle 2 and 3 seen in C, is due to a GPS problem. The issue is thoroughly discussed in Reference 25 [Colour figure can be viewed at wileyonlinelibrary.com]

constraint tightening approach was proposed in Reference 25, with which collision avoidance can be guaranteed even with potentially dangerous constraint violations.

5.2.2 | Intersection-level control loop

To enable comparison, the same experiments were performed using both the converged and 1-step controller. Data from the application of both controllers to one experimental scenario is shown in Figure 10, where the scenario parameters were as in Section 5.1. As the figure illustrates, the difference between the two controllers is very small: the acceleration and velocity profiles in Figure 10B show a high degree of similarity, and the position trajectories in Figure 10A are almost indistinguishable. The differences are most clearly seen in the lower plot of Figure 10C, which shows the changes in t_i^{in} compared with the first coordinated solution at $t = 0$. Here, the smaller but more frequent changes to $\boldsymbol{T}$ by the 1-step controller are clearly differentiated from the less frequent but larger adjustments performed by the converged controller. Note that, while updates are more frequent for the 1-step controller, they are still significantly slower than the vehicle-level update frequency of 10 Hz. The reason is that each adjustment first requires all vehicles to send the relevant information to the central node, which thereafter can solve the QP subproblem (23) and send the updated time-slot back to the vehicles. The process thereby involves the same type of waiting and delays as discussed in Section 5.1.1.

Note also that for both controllers, the initial time-slot schedule $\boldsymbol{T}$ is continuously pushed to later times. This is likely caused by inaccuracies in the prediction model which cause the real system to lag slightly behind the predictions. This explanation is consistent with the nature of the constraint violations discussed in Section 5.2.1, in particular those shown in Figure 9C. The use of a more accurate prediction model is expected to positively affect the behavior.

We want to highlight that the magnitude of the input commands and acceleration as well as the resulting changes in velocity are all small. For comparison, it has been shown that human drivers decelerate with down to $-1.9\ m/s^2$ during intersection approaches without stops (light switching from red to green) and down to $-4.5\ m/s^2$ for solid red lights.[47]

To illustrate the effects of using a simple implementation and deploying the controller on a real system, results from a simulation of both controllers in the same scenario as the experiments are provided in Figure 11. Since one iteration of the SQP requires less than 20 ms in the ideal case, including two-way communication, the 1-step updates

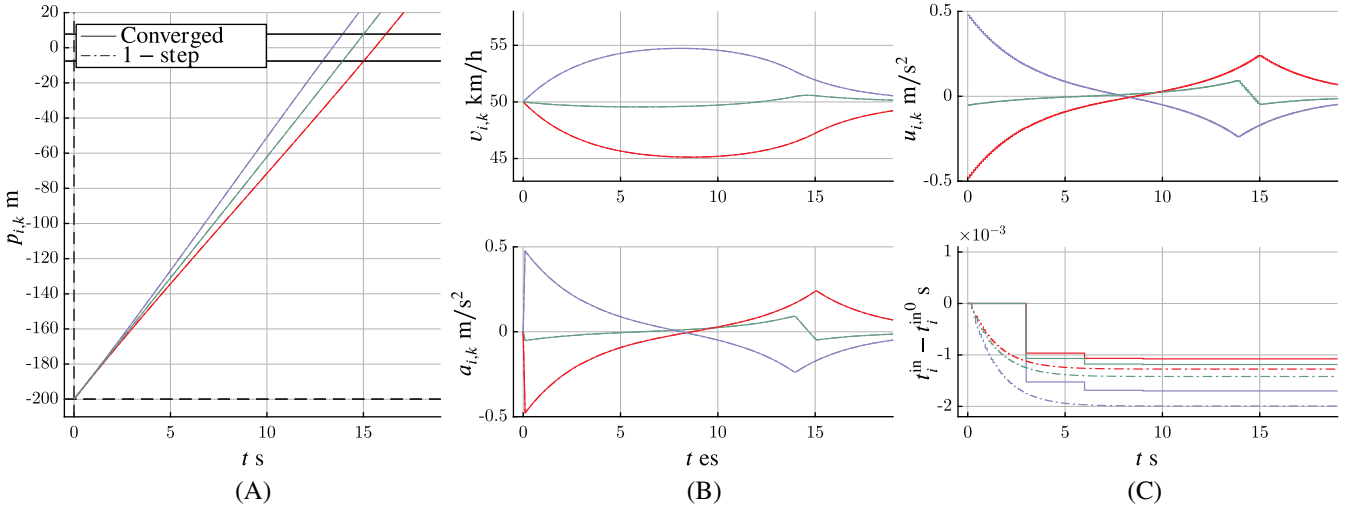


FIGURE 11 Results from simulations where the converged and 1-step intersection controllers were used. Plots as in Figure 10 [Colour figure can be viewed at wileyonlinelibrary.com]

are computed with the same frequency as the vehicle-level controllers (0.1 s) in simulation. The differences between the controllers are even smaller in the simulated case, and are again most noticeable in the control output plot of Figure 11C.

5.2.3 | Rejection of perturbations

To investigate the ability to counteract large perturbations, experimental scenarios were performed where the drivers of the vehicles overrode vehicle-level controllers for short periods of time by pressing the brake or accelerator pedals. Data from two such experiments where the driver of the first vehicle presses the brake pedal is given in Figure 12. For comparison, Figure 12 also shows the open-loop predictions made on the onset of the perturbation, which gives an indication of what the unperturbed trajectories would look like.

Rejection of the perturbations is best handled by both feedback loops, that is, both the action of the vehicles and the time-slot schedule should be adjusted. In particular, when the velocity of the first vehicle is reduced due to the perturbation, the time-slot schedule should be adjusted so that the intersection entry of all vehicles is postponed. Indeed, this is also what occurs in both the Converged and 1-step cases, shown in Figure 12A,B, respectively, where the size of the adjustments in t_i^{in} is shown in the lower plots and the perturbation is represented by the gray slab. Note that since the perturbations are introduced manually by the driver, they differ in length and magnitude between Figure 12A,B.

The benefit of the bi-level control structure is made visible in the middle plots of Figure 12A. Here it is clearly shown that the application of a recomputed time-slot at $t \approx 7$ reduces the magnitude of the control command of vehicle 1 and increases that of vehicle 2, which effectively distributes the effort required to counter the perturbation among the two vehicles.

A similar behavior, however, smaller in magnitude, can be observed for the 1-step controller in Figure 12B, which causes the jagged behavior in the middle plot. Note that as predicted in Section 3.3, the reaction of the intersection-level control loop is delayed for the converged controller, and the large adjustment to $\boldsymbol{T}$ is not performed until 2.5 s after the perturbation. This is due to the relatively long cycle time of the intersection-level control loop, $t_s^O = 3s$, and the use of predicted future states as basis for the SQP, as discussed in Section 3.3.1, which prevents faster reactions.

A perturbed scenario was also simulated to further highlight the benefits of the bi-level control structure and to enable better comparison between the different intersection-level controllers. The result is shown in Figure 13, where we also include the case where the time-slots are not adjusted and a highly idealized, unrealistic controller in which the SQP is solved to convergence every $t_s = 0.1$ s at the current state to serve as benchmarks.

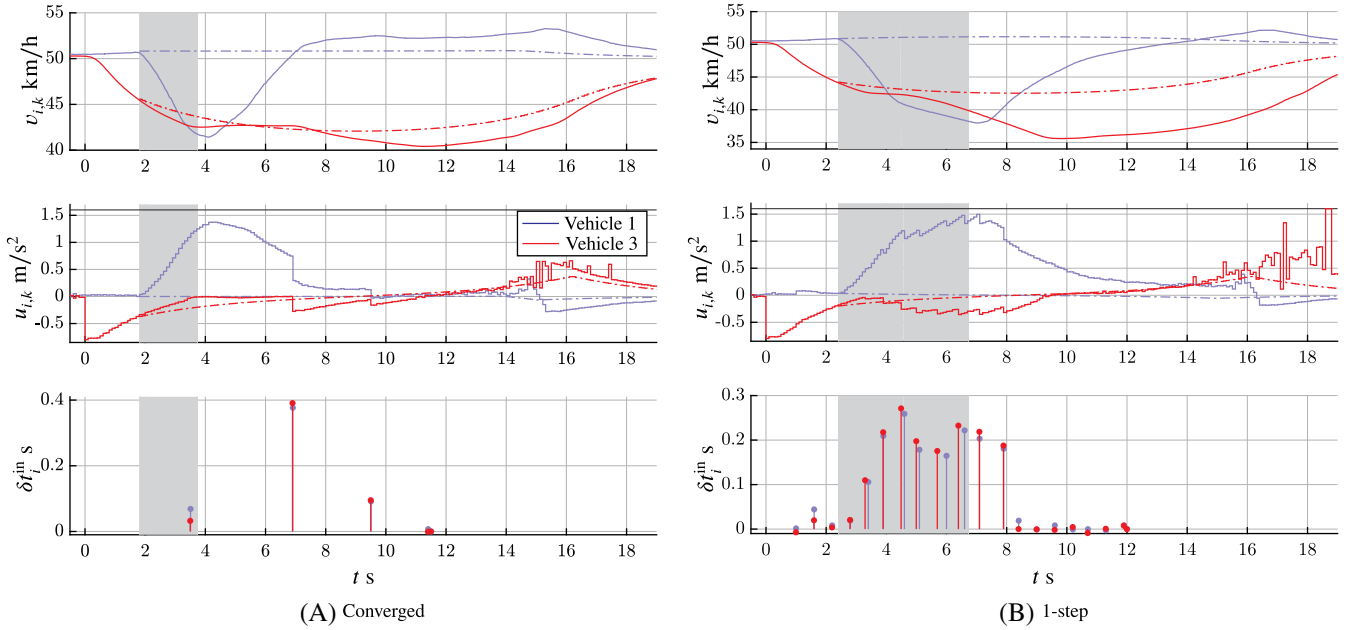


FIGURE 12 Data from experimental scenarios with parameters $Q_1 = 10$, $R_1 = 100$, $Q_2 = Q_3 = 1$, $R_2 = R_3 = 10$ and $v_i^{\text{ref}} = 50\ km/h$ where the driver of the first vehicle press the brake. To ease visualization, the data from Vehicle 2 has been removed from the presentation. A, Shows the response using the converged intersection-level controller and B, the response using the 1-step controller. The plots show from top to bottom the velocity, the acceleration commanded by the vehicle-level MPC and the change between consecutive computations of t_i^{in}, δt_i^{in}. In all plots, the gray bar illustrates the time during which the automated system of the first vehicle is overridden by the driver. The dashed lines in the velocity and acceleration plots shows the MPC's open-loop prediction at the time where the automated system is suspended. The noise in control command of Vehicle 3 around $t \approx 17$ is due to a GPS issue, which is discussed in Reference 25 but omitted here for brevity. MPC, model predictive control [Colour figure can be viewed at wileyonlinelibrary.com]

Note in particular the trajectories corresponding to the case in which the time-slot schedule is kept constant. In this case, the entire effort of perturbation rejection is placed on the first vehicle, with higher transient accelerations and velocities as a consequence. A similar behavior is observed between the time-slot updates when the SQP is solved to convergence every $t_s^O = 3$s. While the effort of rejecting the disturbance is redistributed among the vehicles with recomputed time-slots, large acceleration levels are observed in Vehicle 1 between $t = 3$ and $t = 6$.

With the 1-step controller, the transient velocities and accelerations are lower as the effort to counter the perturbation is continuously distributed between the vehicles. As can be seen in Figure 13B, the size of the time-slot adjustments in the 1-step case is very small, and their application results in no rapid changes in the control command. This is due to the higher update frequency, which allows a gradual adjustment of the intersection-level controller to the disturbance.

Finally, the difference between the positions resulting from the 1-step controller and the idealized case where the SQP solved to convergence every 0.1 s is shown in the lower plot of Figure 13A. As can be seen, the difference is at most in the order of the accuracy of the positioning system. For most of the time, the difference would not be distinguishable from measurement noise. This is a strong indication that there is no major benefit of solving the SQP to a higher accuracy this fast, further motivating the use of the 1-step controller. We note that the corresponding accumulated difference in t_i^{in} is around 50 ms at most, and settles around 1 ms.

Note that the trajectories for the case where the SQP is solved to convergence every $t = 0.1$ s are drawn in all plots of Figure 13, but are indistinguishable from the trajectories corresponding to the 1-step controller.

6 | DISCUSSION AND CONCLUSION

This article addressed the development and experimental validation of a semi-distributed algorithm for optimal coordination of automated vehicles at intersections. In particular, we described a bi-level MPC, where an outer control layer allocates collision free intersection occupancy time-slots by solving a NLP, and a lower control layer computes the OC

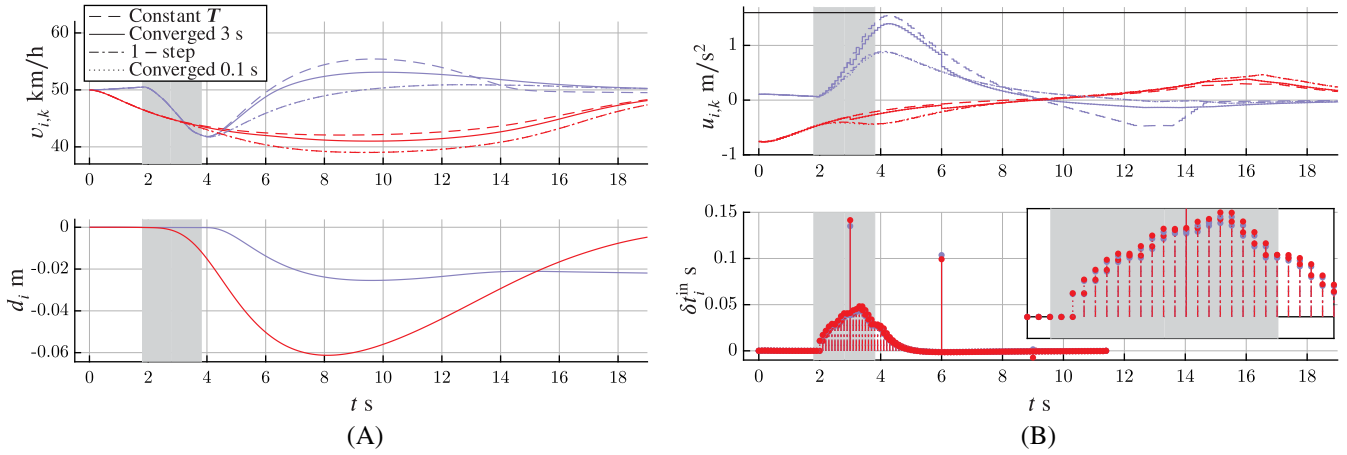


FIGURE 13 Data from simulated scenarios with large perturbations where different versions of the intersection-level controller has been applied. Besides the controllers used in the experiments, the case where the time-slot schedule is kept constant and the case where the SQP is solved to convergence every 0.1 s is shown for comparison. The colors differentiate Vehicle 1 (blue) from Vehicle 3 (red). In the lower plot of A, d_i denotes the difference between the position $p_{i,k}$ resulting from the 1-step controller and the controller where the SQP is solved to convergence every 0.1 s. The perturbation is taken from the experimental scenario of Figure 12A [Colour figure can be viewed at wileyonlinelibrary.com]

commands by solving QPs. We detailed a semi-distributed SQP method used to solve the NLP, and described a practical implementation of the controller, including the integration of the state-of-the art QP solver HPMPC. We demonstrated the applicability of the controller and computational scheme through an extensive experimental campaign. In particular, even though there were substantial imperfections in the implementation, the method was shown to reject large perturbations efficiently and satisfy the collision avoidance constraints to a relevant accuracy.

We want to highlight that the issues observed during the experimental campaign were entirely due to implementation deficiencies and design choices and not inherent to the control formulation or algorithm. On the contrary, the performance observed despite the issues is an indication of the scheme's applicability to real scenarios. For instance, while the ITS-G5 standard has six service channels,[48] and therefore has the potential to let six vehicles communicate simultaneously, larger scenarios would require sequential communication with the current technology. The actual communication time would thereby increase and a delay would be induced: in a scenario with, for example, 42 cars, at least seven rounds where six vehicles communicate in parallel would be required. With the equipment used during the experiment, the wait on Line 12 of Algorithm 1 would increase with at least 24 ms. Even though a larger scenario likely requires more SQP iterations to converge, as indicated by the results of Section 5.1.3, the time required to solve the problem would still be much smaller than that observed in the experiments, and equal or better performance could be expected. It should also be noted that, even in a nonideal communication environment where some vehicles possibly need to resend their data due to packet drops, the added delay is small compared with the one observed during the experiments and would likely not affect the controller performance significantly.

We also want to emphasize the complete parallelizability of the vehicle-level QPs and LPs. Due to this, the time-per-iterate in the SQP will practically be independent of the number of vehicles, and the computational time decided solely by the number of iterates, use of reduced steps, and the time required to solve the QP subproblems. In terms of computation, the algorithm therefore scales well with an increased number of vehicles.

We should also point out that the central part of the algorithm could be performed at a physically remote location, for example, in the "cloud." In this case, the coordinating unit would only be required to provide an access point to the V2V network. Moreover, the ability to function with rather large delays could also motivate the use of cellular communication rather than direct radio-links. With a cellular communication solution, no dedicated intersection infrastructure would be needed at all.

As evidenced by both experimental and simulated results, the bi-level controller successfully managed to handle both large and small perturbations. In particular, by closing the outer control loop, we showed that the controller distributed the effort of rejecting perturbations among all involved vehicles. With such a system, the actuation capacities of all involved vehicles can be used to prevent a collision, should it be necessary, which is an important safety feature. Moreover, the

1-step variation of the intersection-level controller was shown to have comparable performance. Indeed, as indicated by the simulation results, the difference between solving the SQP to full convergence and applying the 1-step scheme are on a scale which makes it irrelevant to the application. As the 1-step scheme requires significantly less communication, faster feedback is thereby enabled in the intersection control loop. While this indicates that the 1-step controller is superior to the Converged controller in the tested cases, further studies are required before general conclusions can be drawn.

6.1 | Future Work

We aim to generalize the method to include scenarios with more than one oncoming lane per road and explicitly account for rear-end collisions between vehicles on the same lane. While rear-end collision avoidance constraints are easy to formulate and include in the centralized setting of Problem (6), they create additional couplings between the vehicles and complicates the application of the decomposition method used in this article. We are currently working on schemes which allows both rear-end collision avoidance constraints and distributed computation.

ACKNOWLEDGEMENTS

This research was funded by The Swedish Research Council under Grant No. 2012-4038, Vinnova under grants 2015-04849 (Copplar project) and 2015-03075 (AstaZero program). The Authors would like to thank Albin Severinson for help with the V2V communication equipment, and express our gratitude toward the following individuals for help with equipment for the experimental validation: Arpit Karsolia and Fredrik von Corswant at Chalmers REVERE-lab, Wojciech Mostovski at Halmstad University, Henrik Lind at Volvo Cars and Alessandro Colombo at Politechnico di Milano. The authors further wish to thank Moritz Diehl for supporting the cooperation between the research groups at Chalmers University and Freiburg University. We also want to thank our colleagues Giuseppe Giordano, Ankit Gupta and Johan Karlsson as well as Marco di Vaio and Gabriel de Campos, who acted as drivers and helpers during the experiments. Finally, we want to thank Fengco Real-Time Control and Leica Geosystems for technical assistance with the hardware.

ORCID

Robert Hult https://orcid.org/0000-0002-1337-3880

REFERENCES

1. Hult R, Campos GR, Steinmetz E, Hammarstrand L, Falcone P, Wymeersch H. Coordination of cooperative autonomous vehicles: toward safer and more efficient road transportation. *IEEE Signal Process Mag*. 2016;33(6):74-84.
2. Colombo A, Del Vecchio D. Efficient algorithms for collision avoidance at intersections. Paper presented at: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control; 2012:145-154.
3. Steinmetz E, Hult R, Campos GR, Wildemeersch M, Falcone P, Wymeersch H. Communication analysis for centralized intersection crossing coordination. Paper presented at: Proceedings of the 11th International Symposium on Wireless Communications Systems (ISWCS); 2014:813-818.
4. Chen L, Englund C. Cooperative intersection management: a survey. *IEEE Trans Intell Transp Syst*. 2016;17(2):570-586.
5. Rios-Torres J, Malikopoulos AA. A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. *IEEE Trans Intell Transp Syst*. 2017;18(5):1066-1077.
6. Dresner K, Stone P. Multiagent traffic management: a reservation-based intersection control mechanism. Paper presented at: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems; 2004:530-537.
7. Dresner K, Stone P. A multiagent approach to autonomous intersection management. *J Artif Intell Res*. 2008;31(1):591-656.
8. Kowshik H, Caveney D, Kumar PR. Provable systemwide safety in intelligent intersections. *IEEE Trans Veh Technol*. 2011;60(3):804-818.
9. Gregoire J, Frazzoli E. Hybrid centralized/distributed autonomous intersection control: using a job scheduler as a planner and inheriting its efficiency guarantees. Paper presented at: Proceedings of the 55th IEEE Conference on Decision and Control (CDC); 2016:2549-2554.
10. Tallapragada T, Cortés J. Coordinated intersection traffic management. *IFAC-PapersOnLine*. 2015;48(22):233-239.
11. Lee J, Park B. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Trans Intell Transp Syst*. 2012;13(1):81-90.
12. Campos GR, Falcone P, Sjöberg J. Autonomous cooperative driving: a velocity-based negotiation approach for intersection crossing. Paper presented at: Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC); 2013:1456-1461.
13. Campos GR, Falcone P, Wymeersch H, Hult R, Sjöberg J. Cooperative receding horizon conflict resolution at traffic intersections. Paper presented at: Proceedings of the 53rd IEEE Conference on Decision and Control (CDC); 2014:2932-2937.

14. Kim KD, Kumar PR. An MPC-based approach to provable system-wide safety and liveness of autonomous ground traffic. *IEEE Trans Autom Control*. 2014;59(12):3341-3356.
15. Miculescu D, Karaman S. Polling-systems-based control of high-performance provably-safe autonomous intersections. Paper presented at: Proceedings of the 53rd IEEE Conference on Decision and Control (CDC); 2014:1417-1423.
16. Qian X, Gregoire J, La Fortelle A, Moutarde F. Decentralized model predictive control for smooth coordination of automated vehicles at intersection. Paper presented at: Proceedings of the European Control Conference (ECC); 2015:3452-3458.
17. Kamal MAS, Imura J, Hayakawa T, Ohata A, Aihara K. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. *IEEE Trans Intell Transp Syst*. 2015;16(3):1136-1147.
18. Murgovski N, Campos GR, Sjöberg J. Convex modeling of conflict resolution at traffic intersections. Paper presented at: Proceedings of the 54th IEEE Conference on Decision and Control (CDC); 2015:4708-4713.
19. Zhang YJ, Malikopoulos AA, Cassandras CG. Optimal control and coordination of connected and automated vehicles at urban traffic intersections. Paper presented at: Proceedings of the American Control Conference (ACC); 2016:6227-6232.
20. Jiang Y, Zanon M, Hult R, Houska B. Distributed algorithm for optimal vehicle coordination at traffic intersections. *IFAC-PapersOnLine*. 2017;50(1):11577-11582.
21. Katriniok A, Kleibaum P, Joševski M. Distributed model predictive control for intersection automation using a parallelized optimization approach. *IFAC-PapersOnLine*. 2017;50(1):5940-5946.
22. Hult R, Zanon M, Gros S, Falcone P. Energy-optimal coordination of autonomous vehicles at intersections. Paper presented at: Proceedings of the European Control Conference; 2018.
23. Hult R., Zanon M., Gros S., Falcone P.. An MIQP-based heuristic for optimal coordination of vehicles at intersections. Paper presented at: Proceedings of the 57th IEEE Conference on Descision and Control; 2018. https://research.chalmers.se/publication/501532/file/501532_Fulltex t.pdf.
24. Hult R, Campos GR, Falcone P, Wymeersch H. An approximate solution to the optimal coordination problem for autonomous vehicles at intersections. Paper presented at: Proceedings of the American Control Conference (ACC); 2015:763-768.
25. Hult R, Zanon M, Gros S, Falcone P. Optimal coordination of automated vehicles at intersections: theory and experiments; 2018. https://arixiv.2018.
26. Hult R, Zanon M, Gros S, Falcone P. Primal decomposition of the optimal coordination of vehicles at traffic intersections. Paper presented at: Proceedings of the 55th IEEE Conference on Decision and Control (CDC); 2016:2567-2573.
27. Zanon M, Gros S, Wymeersch H, Falcone P. An asynchronous algorithm for optimal vehicle coordination at traffic intersections. *IFAC-PapersOnLine*. 2017;50(1):12008-12014.
28. Zanon M, Hult R, Gros S, Falcone P. Experimental validation of distributed optimal vehicle coordination. Paper presented at: Proceedings of the European Control Conference; 2018.
29. Frison G, Sorensen HB, Dammann B, Jorgensen JB. High-performance small-scale solvers for linear model predictive control. Paper presented at: Proceedings of the European Control Conference (ECC); 2014:128-133.
30. Diehl M. Real-Time Optimization for Large Scale Nonlinear Processes. vol. 920 of Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik: Düsseldorf: VDI Verlag; 2002. http://www.ub.uni-heidelberg.de/archiv/1659/.
31. Hult R, Zanon M, Gros S and Falcone P. Optimal Coordination of Automated Vehicles at Intersections with Turns, in Proceedings of the European Control Conference (ECC); 2018.
32. Nocedal J, Wright SJ. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. 2nd ed. New York, NY: Springer; 2006.
33. Büskens C, Maurer H. Ch: sensitivity analysis and real-time optimization of parametric nonlinear programming problems. *Online Optimization of Large Scale Systems*. Berlin Heidelberg / Gemany: Springer; 2001:3-16.
34. Diehl M, Findeisen R, Schwarzkopf S, et al. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part I: description of the method. *Automatisierungstechnik*. 2002;50(12):557-567.
35. Findeisen R, Allgöwer F. Computational delay in nonlinear model predictive control. Paper presented at: Proceedings of the International Symposium on Advanced Vehicle Control of Chemical Processes, ADCHEM; 2003.
36. Chen W, Ballance DJ, O'Reilly J. Model predictive control of nonlinear systems: computational delay and stability. *IEEE Trans Autom Control*. 2000;147(4):387-394.
37. Fletcher R. *Practical Methods of Optimization*. 2nd ed. Chichester, NH: Wiley; 1987.
38. Frison G, Kouzoupis D, Sartor T, Zanelli A, Diehl M. BLASFEO: basic linear algebra subroutines for embedded optimization; 2017. arXiv:1704.02457.
39. Axehill D. Controlling the level of sparsity in MPC. *Syst Control Lett*. 2015;76:1-7.
40. Frison G, Kouzoupis D, Jørgensen JB, Diehl M. An efficient implementation of partial condensing for nonlinear model predictive control. Paper presented at: Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC); 2016:4457-4462; IEEE.
41. Diehl M, Ferreau HJ, Haverbeke N. Ch: nonlinear model predictive control lecture notes in control and information sciences. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*. Vol 384. New York, NY: Springer; 2009:391-417.
42. Hult R, Sancar E, Jalalmaab M, et al. Design and experimental validation of a cooperative driving control architecture for the grand cooperative driving challenge 2016. *Trans Intell Transp Syst*. 2017;19(4):1290-1301.
43. RENDITS. http://www.rendits.com/. Accessed. May 11, 2016.
44. Zanon M, Gros S, Diehl M. A tracking MPC formulation that is locally equivalent to economic MPC. *J Process Control*. 2016;45:30-42.

45. Hult R, Zanon M, Gros S, Falcone P. Optimal coordination of three cars approaching an intersection. https://youtu.be/nYSXvnaNRK4. Accessed. March 03, 2017.
46. Fernandez JA, Borries K, Cheng L, Kumar BVK, Stancil DD, Bai F. Performance of the 802.11p physical layer in vehicle-to-vehicle environments. *IEEE Trans Veh Technol*. 2012;61(1):3-14.
47. Rittger L, Schmidt G, Maag C, Kiesel A. Driving behaviour at traffic light intersections. *Cogn Tech Work*. 2015;17(4):593-605.
48. European Telecommunications Standards Institute. ETSI EN 302 663: access layer specification for intelligent transport systems operating in the 5 GHz frequency band; 2012.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Hult R, Zanon M, Frison G, Gros S, Falcone P. Experimental validation of a semi-distributed sequential quadratic programming method for optimal coordination of automated vehicles at intersections. *Optim Control Appl Meth*. 2020;41:1068–1096. https://doi.org/10.1002/oca.2592