Preprints, 5th IFAC Conference on
Analysis and Control of Chaotic Systems
Eindhoven, The Netherlands, Oct 30 - Nov 1, 2018

Design of a supervisory controller for Cooperative Intersection Control using Model Predictive Control

Falco Creemers * **Alejandro Ivan Morales Medina** *
Erjen Lefeber * **Nathan van de Wouw** *,⋆

* *Department of Mechanical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (e-mail: F.M.G.Creemers@alumnus.tue.nl, A.I.Morales.Medina@tue.nl, A.A.J.Lefeber@tue.nl, N.v.d.Wouw@tue.nl).*

Abstract: The Cooperative Intersection Control (CIC) methodology ensures a safe, smooth traffic flow through an automated intersection by means of virtual platooning. In this paper, we address the design of a centralised supervisory controller for CIC which optimises the crossing sequence of vehicles. We propose an approach in which the intersection as a whole is modelled as a hybrid system, which evolves in both continuous-time and in discrete-time. This hybrid system model resembles a queueing system, and relates the entry of vehicles into the intersection to a measure of the delay of their travel through the intersection. We design a supervisory controller using Model Predictive Control (MPC), which aims to minimise the vehicles' average delay by controlling their access to the intersection. A simulation study based on real-life data demonstrates the effectiveness of the MPC approach compared to a first-come-first-served (FCFS) policy and a conventional traffic light controller. This study shows that MPC achieves a faster transient response and a lower average delay, thereby increasing the throughput of the intersection.

Keywords: Hybrid systems, Model Predictive Control, Autonomous vehicles, Cooperative Intersection Control, Traffic control.

1. INTRODUCTION

The rapid increase in motorisation and urbanisation has led to more and more traffic congestion, which, in turn, results in increased delays in the transportation of people and goods. Typically, traffic congestion occurs at road intersections, since these are the places where the vehicles' paths cross. Currently, access to an intersection is regulated through stop signs, traffic lights, etc., each of which require vehicles to wait for their turn, thereby limiting the throughput of the intersection.

Recent advances in vehicle automation and wireless communication are expected to improve both the safety and efficiency of intersections. Various solutions to automate an intersection have been proposed in recent years. For example, a reservation-based method is proposed in Dresner and Stone (2008), where vehicles can reserve regions in the intersection for certain time slots, which an intersection manager accepts on a first-come-first-served (FCFS) basis. Other approaches focus on planning the trajectories of the vehicles such that all vehicles can cross the intersection safely. For instance, a scheduling-based method is proposed in Colombo and Vecchio (2015), where the set of control actions which avoid a collision is determined. Other works, such as Qian et al. (2015); Hult et al. (2016), use optimal control techniques to coordinate the movement of the vehicles through the intersection. The approach to intersection management that this paper is focused on is the Cooperative Intersection Control (CIC) methodology proposed in Morales Medina et al. (2015, 2018). It is active in a region around the intersection, called the Cooperation Zone (CZ), and relies on so-called Virtual Cooperative Adaptive Cruise Control (VCACC) to allow vehicles to form virtual platoons of vehicles in the intersection. The term 'virtual' refers to the fact that vehicles travelling along different trajectories can also form a platoon. The methodology is split into two levels: a supervisory level which determines a vehicle's access priority, which is currently done on a FCFS basis, and an execution level which controls the longitudinal inter-vehicle dynamics.

Many of the proposed intersection management solutions focus on controlling and optimising the dynamics of the vehicles' movement through the intersection. However, optimisation of the crossing sequence is typically not addressed and the crossing sequence is often pre-defined or is determined on a FCFS basis. The performance of an automated intersection can be improved by optimising the access protocol determining the crossing sequence.

⋆ N. van de Wouw is also affiliated with the Civil, Environmental & Geo-Engineering Department, University of Minnesota, U.S.A. and with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

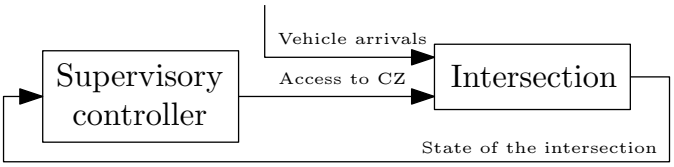

Fig. 1. The system representation used to optimise the crossing sequence.

The current paper presents a Model Predictive Control (MPC) strategy to optimise the crossing sequence. MPC relies on dynamic models of the system to be optimised while satisfying given constraints and is often used in industrial processes such as power generation or in chemical plants. In these applications, the controlled variables are typically continuous-valued, whereas the supervisory controller controls the sequence of discrete vehicles. Additionally, logical relationships among the states of the system determine the timing of the vehicle sequence. This affects the implementation of the system, which is described for the scheduling of industrial process in Gallestey et al. (2003), and Creemers (2017) describes the implementation of the supervisory controller.

The main contribution of this paper is the design of a centralised supervisory controller which optimises the crossing sequence within the CIC methodology of Morales Medina et al. (2015, 2018). This design is based on a model abstraction of the intersection as a queueing system, which is formulated in the hybrid systems framework proposed in Goebel et al. (2009, 2012). This model evolves both in discrete-time and in continuous-time, which allows for a description using integer queue lengths while also describing the exact time between any two vehicle arrivals or departures at/into the intersection. The intersection model gives a measure of the vehicles' average delay, and is used in the design of an MPC strategy. The resulting controller is able to minimise the vehicles' average delay while also respecting access timing constraints which ensure a safe crossing of the vehicles through the intersection.

The outline of this paper is as follows. The intersection is modelled as a hybrid system in Section 2. The control problem is formalised in Section 3. Section 4 describes the design of the supervisory controller. In Section 5, a simulation study is presented to show the effectiveness of the designed controller. This paper concludes in Section 6.

2. INTERSECTION MODEL

In CIC, the supervisory level and the execution level jointly control the behaviour of the vehicles in the CZ, as described in Morales Medina et al. (2015, 2018). To optimise the crossing sequence, a simplified system representation is used, which is shown in Fig. 1. Here, the 'intersection' block includes the whole of the Cooperation Zone (CZ) and the dynamics of the vehicles in it, which are controlled by the execution-level controller, which guarantees a (virtual) distance between vehicles in the intersection forming a (virtual) platoon. The intersection is abstracted as a queueing system, where vehicles wait in queues outside the CZ, while the supervisory controller manages their access into the CZ. The CZ and the vehicles waiting in queues are schematically depicted in Fig. 2.

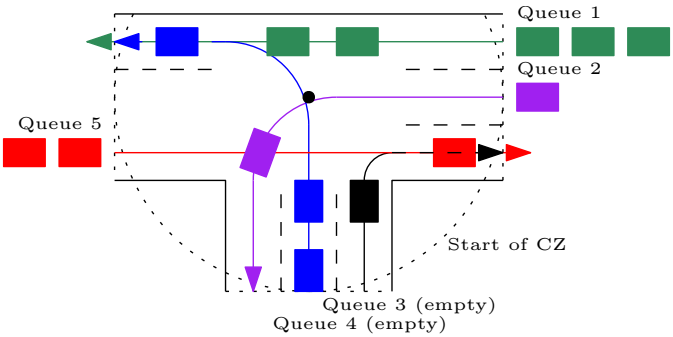

Fig. 2. A schematic depiction of the Cooperation Zone (CZ) showing vehicles waiting in queues outside the CZ or travelling through the CZ. Based on intersection S4 of Fleuren and Lefeber (2016).

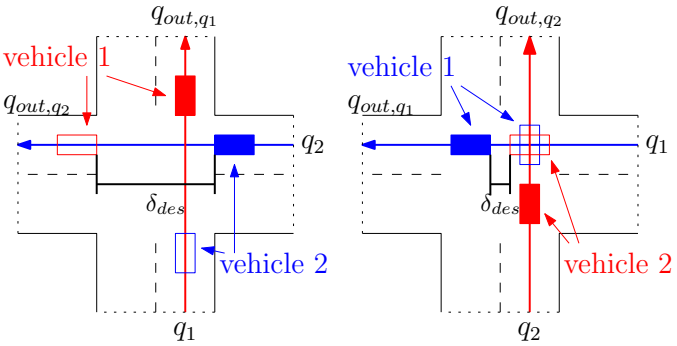

Fig. 3. Two situations with different vehicle sequences, changing the desired virtual inter-vehicle distances δ_{des}. The coloured boxes give the current positions of each vehicle. The outlined boxes are projections of the other vehicle onto the vehicles' own trajectory.

The intersection model aims to describe the arrivals and departures of individual vehicles at their queues. A safe crossing is ensured by enforcing a service time between any two subsequent vehicle departures into the CZ.

2.1 Service time

The service time T^s is defined through the virtual inter-vehicle distance δ_{des} of the first vehicle in a queue, which is guaranteed by the execution-level controller to ensure a vehicle's safe crossing. This distance varies with the trajectories and sequence of the vehicles, as depicted in Fig. 3. For general intersections, a vehicle's trajectory is determined by its incoming queue and its exit lane. This paper considers a simplified intersection where each queue has a single corresponding exit lane, which ensures that the service time of all queues does not vary over time due to vehicles in a queue having different trajectories.

Then the virtual inter-vehicle distance between a departure from queue $q_1 \in \mathcal{Q} = \{1, \ldots, n_l\}$, with n_l the number of queues, and a subsequent departure from queue $q_2 \in \mathcal{Q}$ is defined as a function of the queues of those vehicles $\delta_{des}(q_1, q_2)$. This calculation is based on the layout of the intersection and the inter-vehicle spacing policy, discussed in detail in Morales Medina et al. (2015, 2018). The service time $T^s_{q_1,q_2}$ between two vehicles is then given by

$$T^s_{q_1,q_2} = \frac{\delta_{des}(q_1, q_2,)}{V}, \tag{1}$$

with V the vehicle velocity and where $T^s_{q_1,q_2} = 0$ if the trajectories of the vehicles do not intersect. The velocity V is assumed to be constant, which can be (approximately) guaranteed by the execution-level controller.

2.2 Hybrid system model

The intersection is abstracted as a queueing system which considers the arrival and departure of individual vehicles at/from queues. An accurate description of the queueing system dynamics urges both a continuous-time and a discrete-time description. Namely, arrivals and departures may not occur precisely at the sampling instants of a discretised model, and enforcing the service times requires keeping track of the continuous-time elapsed between departures into the intersection. On the other hand, a discrete-time model easily describes integer queue lengths. Therefore, the intersection is modelled as a hybrid system in the framework of Goebel et al. (2009, 2012). Such a system evolves in hybrid time (t, j) and is of the form

$$\dot{z} = f(z, u) \text{ for } z \in \mathcal{C}, \quad z^+ = g(z, u) \text{ for } z \in \mathcal{D}, \tag{2}$$

where $z \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, f is the flow map, g is the jump map, $\mathcal{C}$ is the flow set, $\mathcal{D}$ is the jump set and $z^+ = z(t, j + 1)$. The arc that the system follows through the hybrid time domain (t, j) is called the solution ϕ.

The evolution of the intersection in hybrid time is governed by the arrival or departure of individual vehicles. Both are modelled as instantaneous events which make the jump set $\mathcal{D}$ active. Otherwise, the flow set $\mathcal{C}$ is active. Arrivals to queue $q \in \mathcal{Q}$ are given by an external disturbance $w^a_q(t)$, and departures are given by the control input $u_q(t)$. They are modelled as

$$u_q(t) = \begin{cases} 1 & \text{if a vehicle departs from queue } q \text{ at time } t, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

with vector form $u(t) = [u_1(t), \dots, u_{n_l}(t)]^\top \in \{0, 1\}^{n_l}$ and $w^a(t)$ defined analogously. Being external inputs, both signals are defined solely in continuous time. These inputs are defined in the hybrid time domain as being in the domain of the solution, i.e. for all $(t, j) \in \text{dom } \phi$, as

$$u(t, j) = \begin{cases} u(t), & j = \min\{j^* | (t, j*) \in \text{dom } \phi\}, \\ 0, & j \neq \min\{j^* | (t, j*) \in \text{dom } \phi\}, \end{cases} \tag{4}$$

and $w^a(t, j)$ defined analogously. This definition ensures that each unique event in continuous time corresponds with a unique event in hybrid time.

Before defining the sets $\mathcal{C}$ and $\mathcal{D}$, a minimum inter-event time must be enforced to prevent the occurrence of Zeno behaviour, in which an infinite number of jumps occur in a finite time interval. This requires the following assumption.

Assumption 1. $\forall t \geq 0$ s.t. $u_q(t) = 1$ for some queue q, there exists an $\varepsilon_u > 0$ such that $u_q(t', j') = 0 \ \forall t' \in (t, t + \varepsilon_u]$ and $(t', j') \in \text{dom } \phi$. Additionally, $\forall t \geq 0$ s.t. $w^a_q(t) = 1$ for some queue q, there exists an $\varepsilon_w > 0$ such that $w^a_q(t', j') = 0 \ \forall t' \in (t, t + \varepsilon_w]$ and $(t', j') \in \text{dom } \phi$.

The validity of this assumption will be enforced in the definition of the flow and jump sets through the use of inter-event timers. The minimum inter-departure time ε_u will be guaranteed through a constraint of the MPC controller. The minimum inter-arrival time ε_w is trivially satisfied for small ε_w given the bounded vehicle velocity V and the finite, strictly positive length of a vehicle.

An inter-departure timer $t_{c,q}(t, j)$ and an inter-arrival timer $t_{w,q}(t, j)$ are defined for every queue $q \in \mathcal{Q}$ which count the time since the last departure/arrival at each queue. Their dynamics are given by

$$\dot{t}_{c,q}(t, j) = 1 \tag{5}$$

$$t_{c,q}(t, j + 1) = \begin{cases} t_{c,q}(t, j) & \text{if } u_q(t, j) = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

with vector form $t_c(t, j) = [t_{c,1}(t, j), \dots, t_{c,n_l}(t, j)]^\top \in \mathbb{R}^{n_l}_{\geq 0}$. The dynamics for $t_{w,q}(t, j)$ are defined analogously.

Using these timers, the flow and jump sets are defined such that the system only jumps in discrete time if both 1) an event occurs at some queue, and 2) the inter-event timer of that queue has passed the minimum inter-event time. If the minimum inter-event time has not passed, the event is not valid and the system continues to flow in continuous time. Otherwise, a vehicle could be allowed into the intersection too early, potentially risking a collision. To link arrivals with $t_w(t, j)$ and link departures with $t_c(t, j)$, two flow sets and two jump sets are defined. The flow and jump set $\mathcal{C}_u, \mathcal{D}_u$ dependent on vehicle departures are given by

$$\begin{aligned} \mathcal{C}_u = &\{u(t, j) = \underline{0}^{n_l}\} \times \left\{t_c(t, j) \in \mathbb{R}^{n_l}_{\geq 0}\right\} \\ &\bigcup \{u(t, j) \in \{0, 1\}^{n_l}\} \setminus \{\underline{0}^{n_l}\} \\ &\times \{t_c(t, j) \mid \exists q \in \mathcal{Q} \text{ s.t. } u_q(t, j) = 1 \wedge t_{c,q}(t, j) < \varepsilon_u\}, \\ \mathcal{D}_u = &\{u(t, j) \in \{0, 1\}^{n_l}\} \setminus \{\underline{0}^{n_l}\} \\ &\times \{t_c(t, j) \mid t_{c,q}(t, j) \geq \varepsilon_u \forall q \in \mathcal{Q} \text{ s.t. } u_q(t, j) = 1\}, \end{aligned} \tag{7}$$

where $\underline{0}^{n_l}$ is the zero vector of length n_l. The flow and jump sets dependent on vehicle arrivals $\mathcal{C}_w, \mathcal{D}_w$ are defined analogously using $w^a(t, j), t_w(t, j)$ and ε_w.

With these sets, the system flows in continuous-time if both $\mathcal{C}_u$ and $\mathcal{C}_w$ are active, and jumps in discrete-time if $\mathcal{D}_u$ or $\mathcal{D}_w$ are active. This formalised by the conditions

$$(u(t, j), t_c(t, j)) \in \mathcal{C}_u \wedge (w^a(t, j), t_w(t, j)) \in \mathcal{C}_w, \tag{8}$$

$$(u(t, j), t_c(t, j)) \in \mathcal{D}_u \vee (w^a(t, j), t_w(t, j)) \in \mathcal{D}_w. \tag{9}$$

The remaining system dynamics involve two states. First, the dynamics of the queue lengths $x_q(t, j)$ are given by

$$\begin{aligned} \dot{x}_q(t, j) &= 0, \\ x_q(t, j + 1) &= x_q(t, j) + w^a_q(t, j) - u_q(t, j), \end{aligned} \tag{10}$$

with vector form $x(t, j) = [x_1(t, j), \dots, x_{n_l}(t, j)]^\top \in \mathbb{N}^{n_l}_0$.

The final state is the queue activity $m_q(t, j) \in \{0, 1\}$. When a vehicle leaves a queue, that queue becomes active to signal that a vehicle has recently left that queue. A queue then becomes inactive once the inter-departure time $t_{c,q}(t, j)$ has passed the maximum service time. Thus, the hybrid dynamics of $m_q(t, j)$ are given by

$$\begin{aligned} \dot{m}_q(t, j) &= 0, \\ m_q(t, j + 1) &= \begin{cases} 1 & \text{if } u_q(t, j) = 1, \\ 0 & \text{if } t_{c,q}(t, j + 1) \geq \max\limits_{i \in \mathcal{Q} | \{q\}} T^s_{q,i}, \\ m_q(t, j) & \text{otherwise,} \end{cases} \end{aligned} \tag{11}$$

with $m(t, j) = [m_1(t, j), \dots, m_{n_l}(t, j)]^\top \in \{0, 1\}^{n_l}$.

The full state of the system is given by

$$z(t,j) = \left[x(t,j)^\top \; t_c(t,j)^\top \; t_w(t,j)^\top \; m(t,j)^\top\right]^\top, \quad (12)$$

and the full hybrid dynamics are described by

$$\begin{aligned}
\dot{x}_q(t,j) &= 0,\\
\dot{t}_{c,q}(t,j) &= 1,\\
\dot{t}_{w,q}(t,j) &= 1,\\
\dot{m}_q(t,j) &= 0,
\end{aligned}$$

$$\begin{aligned}
x_q(t,j+1) &= x_q(t,j) + w^a(t,j) - u(t,j),\\
t_{c,q}(t,j+1) &= \begin{cases} t_{c,q}(t,j) & \text{if } u_q(t,j) = 0,\\ 0 & \text{otherwise,}\end{cases}\\
t_{w,q}(t,j+1) &= \begin{cases} t_{w,q}(t,j) & \text{if } w^a_q(t,j) = 0,\\ 0 & \text{otherwise,}\end{cases}\\
m_q(t,j+1) &= \begin{cases} 1 & \text{if } u_q(t,j) = 1,\\ 0 & \text{if } t_{c,q}(t,j+1) \geq \max\limits_{i\in\mathcal{Q}|\{q\}} T^s_{q,i},\\ m_q(t,j) & \text{otherwise,}\end{cases}
\end{aligned} \quad (13)$$

for every queue $q \in \mathcal{Q}$. The continuous-time dynamics in (13) are active if the flow condition (8) is satisfied, and the discrete-time dynamics are active if the jump condition (9) is satisfied, as in the standard form of (2).

Finally, the measured output of the system $y(t,j)$ is equal to the state of the system $z(t,j)$, which is given by

$$y(t,j) = z(t,j). \quad (14)$$

As shown in Fig. 1, this output is used solely by the controller to optimise the crossing sequence.

3. PROBLEM STATEMENT

We consider a general intersection model described by (13) for every queue $q \in \mathcal{Q}$. The problem to be solved is to design a control strategy which minimises the delay that the vehicles in the queues experience, while respecting the service times between vehicle departures into the CZ. The objective function to be minimized and the constraints to be respected are described in more detail in the next section. We assume that the designed controller is centralised using roadside infrastructure and has full knowledge of the state of the intersection.

4. CONTROLLER DESIGN

This section discusses the design of the supervisory controller for the hybrid system. This controller must optimise the control input $u(t,j)$ so as to minimise the average delay of the vehicles in the intersection. At the same time, the service times must be respected between any two vehicle departures, which acts as a set of constraints on the control input. Given the combination of an optimisation objective and constraints, a Model Predictive Control (MPC) approach is used. MPC predicts the behaviour of the controlled system, subject to given constraints, over a finite continuous-time horizon. It then selects the control actions that minimise a given cost function.

As there is little research on MPC for hybrid systems, a discrete-time MPC controller is used which samples the system and uses a discretised version of (13) in its predictions. To achieve a feedback loop, the controller only implements the first step of its optimal input trajectory, repeating its optimisation every time-step.

For the sake of brevity, the discretised system is not presented in this paper; it can be found in Creemers (2017). Here, the discretised system is written as

$$\begin{aligned}
z(k+1) &= F(z(k), w^a(k), u(k)),\\
y(k) &= z(k).
\end{aligned} \quad (15)$$

To match the definitions of $w^a_q(t), u_q(t)$ in (3), $w^a_q(k)$ and $u_q(k)$ are equal to 1 if there is a vehicle arrival/departure in the interval $(t_{k-1}, t_k]$, and zero otherwise. This requires Δt to be such that there is at most a single arrival or departure at each queue between two sampling instances. This can be guaranteed by choosing $\Delta t \leq \max\{\varepsilon_u, \varepsilon_w\}$, which also ensures that $w^a(k)$ and $u(k)$ remain in the flow and jump sets as defined in (7).

The remaining components of the problem setting for the MPC controller design are the cost function, the constraints, and a disturbance model predicting the arrival of vehicles, which are presented next.

4.1 Constraints

It is necessary to place constraints on the control input to enforce the service times between any two subsequent vehicle departures. Otherwise, unsafe situations could result if the service times are not respected.

The service times are enforced through two sets of constraints. First, assume that a vehicle in queue q_2 follows a vehicle which has departed from some queue q_1, with $q_1, q_2 \in \mathcal{Q}$. Then the vehicle from queue q_2 may not depart from its queue if the inter-departure time of some active queue q_1 is smaller than $T^s_{q_1,q_2}$, i.e., $t_{c,q_1} < T^s_{q_1,q_2}$. If a queue is inactive, then the service times of that queue have, by definition, passed and that queue does not need to be taken into account. On the other hand, if $t_{c,q_1} \geq T^s_{q_1,q_2}$ for all active queues, the vehicle in queue q_2 may depart into the CZ. These constraints are formulated as

$$\begin{aligned}
u_{q_2}(k) &= 0 \qquad \text{if } \exists q_1 \in \mathcal{Q} \text{ s.t.}\\
&\qquad\qquad \left(m_{q_1}(k) = 1 \wedge t_{c,q_1}(k) < T^s_{q_1,q_2}\right),\\
u_{q_2}(k) &\in \{0,1\} \quad \text{if } \nexists q_1 \in \mathcal{Q} \text{ s.t.}\\
&\qquad\qquad \left(m_{q_1}(k) = 1 \wedge t_{c,q_1}(k) < T^s_{q_1,q_2}\right),
\end{aligned} \quad (16)$$

for every $q_2 \in \mathcal{Q}$ with respect to all other queues $q_1 \in \mathcal{Q}$.

The constraints of (16) constrain vehicle departures based on previous departures. The second set of constraints deals with the fact that multiple vehicles could depart at the same time if $u_{q_2} \in \{0,1\}$ for more than a single queue q_2. Let there be two vehicles from queues q and i who wish to depart at the same time, with $q, i \in \mathcal{Q}$. This can only be permitted if the trajectories of those vehicles do not cross, in which case $T^s_{q,i} = T^s_{i,q} = 0$. This constraint is given by

$$\begin{bmatrix} u_q(k) \\ u_i(k) \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ if } T^s_{q,i} > 0, \quad (17)$$

which must hold for all pairs $q, i \in \mathcal{Q}$ such that $i > q$. It does not need to be defined for all pairs $q, i \in \mathcal{Q}$, since this would create multiple identical (redundant) constraints.

The notation of the input constraints is shortened through the use of a state-dependent input set $\mathcal{U}(m(k), t_c(k))$,

which follows from (16) and (17). Then the constraints on the control input are written as

$$u(k) \in \mathcal{U}(m(k), t_c(k)). \tag{18}$$

4.2 Disturbance model

The disturbance model is used to predict the arrival of vehicles. Without it, the controller would base its optimisation solely on the current queue lengths without taking into account possible differences in the vehicle arrival rates of those queues. The disturbance model employs deterministic uniform vehicle inter-arrival times with an average arrival rate of λ_q vehicles per second for queue $q \in \mathcal{Q}$. The predicted arrival of a vehicle is given by

$$\hat{w}_q^a(k) = \begin{cases} 1 & \text{if } t_{w,q}(k) \geq \lambda_q^{-1}. \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Given the stochastic and time-varying nature of the actual arrival process, the current average arrival rate $\lambda_q(k)$ is estimated using a moving average window as

$$\lambda_q(k) = \frac{1}{\Delta t N_w} \sum_{i=0}^{N_w} w_q^a(k-i), \tag{20}$$

with $w_q^a(k) = 0$ for $k < 0$, and $\lambda(k) = [\lambda_1(k), \ldots, \lambda_{n_l}(k)]^\top$. The moving average window makes it possible for the controller to deal with variations in the arrival rate, provided that this variation occurs over a time-span longer than the window length N_w.

The notation of the disturbance model is shortened as

$$\hat{w}^a(k) = F_w(t_w(k), \lambda(k)). \tag{21}$$

Although this disturbance model does not fully capture the real-world arrival process, which is typically stochastic, the prediction horizon is generally short enough so that few vehicle arrivals will occur in this time, and the prediction does not become too inaccurate.

4.3 MPC problem formulation

The full MPC optimisation problem is now given by

$$\begin{aligned} \min_{u(k),\ldots,u(k+N_p-1)} \quad & J(k) = \sum_{i=0}^{N_p-1} c^\top x(k+i), \\ \text{s.t.} \quad & \\ & z(k+i+1) = F(z(k+i), \hat{w}^a(k+i), u(k+i)) && \forall i \in \mathcal{N}, \\ & x(k+i) \in \mathbb{N}_0^{n_l} && \forall i \in \mathcal{N}, \\ & t_c(k+i), t_w(k+i) \in \mathbb{R}_{\geq 0}^{n_l} && \forall i \in \mathcal{N}, \\ & m(k+i) \in \{0,1\}^{n_l} && \forall i \in \mathcal{N}, \\ & u(k+i) \in \mathcal{U}(m(k+i), t_c(k+i)) && \forall i \in \mathcal{N}, \\ & \hat{w}^a(k+i) = F_w(t_w(k+i), \lambda(k)) && \forall i \in \mathcal{N}. \end{aligned} \tag{22}$$

with prediction steps $\mathcal{N} = \{0, \ldots, N_p - 1\}$. The cost function is chosen such to minimise the average delay of the vehicles arriving at the intersection. A measure of the average delay is given by the queue lengths, since the only delay that vehicles experience is the time that they spend in the queues. Thus, a linear cost function with weighting coefficients c is chosen. This weighs every vehicle in each queue equally and is chosen to prevent the controller from favouring long queues while neglecting shorter queues with low arrival rates. Additionally, note that a linear cost function in itself does not cause instability because the queue lengths are non-negative.

After solving the optimisation problem, the controller provides only the first element of its optimal input trajectory, $u(k)$. The discrete-time signal is converted to continuous-time in the form of (3) as

$$u_q(t) = \begin{cases} 1 & \text{if } u_q(k) = 1 \text{ at time } t = t_k, \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

which ensures that vehicles only depart at the sampling instances of the controller, between which $u_q(t) = 0$.

5. SIMULATION RESULTS

To show the benefit of the MPC approach, we compare its performance to the results of a vehicle-actuated traffic light, which empties each queue in some cycle, and of the FCFS policy currently used in Morales Medina et al. (2015, 2018). Simulations are performed in Simulink, where the hybrid system is implemented using the Hybrid Equations Toolbox by Sanfelice et al. (2014), and the controller is implemented using HYSDEL3 by Torrisi et al. (2000) and the Multi-Parametric Toolbox 3 by Herceg et al. (2013).

The simulations are performed using the real-life data of intersection S4 of Fleuren and Lefeber (2016). This intersection has five queues, each with its own unique trajectory as depicted in Fig. 2. To better compare the behaviour of each controller, vehicles arrive according to deterministic inter-arrival times with arrival rates equal to

$$\lambda = \left[\tfrac{370}{3600} \;\; \tfrac{164}{3600} \;\; \tfrac{194}{3600} \;\; \tfrac{167}{3600} \;\; \tfrac{705}{3600}\right]^\top \tag{24}$$

vehicles per second.

The simulations are performed using two sets of service times: for human drivers and for vehicles automated through CIC. This allows for a comparison of the current state-of-practice, which is a traffic light with human drivers, with vehicles automated through CIC using either an FCFS policy or the MPC controller proposed here to regulate access of the automated vehicles into the intersection. The service times for human drivers are equal to

$$T^s_{\text{human}} = \begin{bmatrix} 1.9 & 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 5 & 5 \\ 0 & 0 & 2.2 & 0 & 5 \\ 6 & 4 & 0 & 2 & 5 \\ 0 & 5 & 4 & 4 & 2 \end{bmatrix}. \tag{25}$$

The service times for automated vehicles are based on the desired inter-vehicle distances in CIC. In Morales Medina et al. (2015, 2018), this is given by $\delta_{des} = r + hV$, with r the stand-still inter-vehicle distance in meters and h the headway time in seconds. For subsequent vehicles coming from the same queue, these are $r = 8$ and $h = 0.3$. For subsequent vehicles coming from different queues, they are $r = 10$ and $h = 0.5$. With $V = 30$ km/h, this gives

$$T^s_{\text{CIC}} = \begin{bmatrix} 1.26 & 0 & 0 & 1.7 & 0 \\ 0 & 1.26 & 0 & 1.7 & 1.7 \\ 0 & 0 & 1.26 & 0 & 1.7 \\ 1.7 & 1.7 & 0 & 1.26 & 1.7 \\ 0 & 1.7 & 1.7 & 1.7 & 1.26 \end{bmatrix}. \tag{26}$$

Table 1 gives the cycle of the traffic light. The MPC controller uses a prediction horizon of $N_p = 35$ time

Table 1. The cycle of the traffic light.

Mode	Queues green	Green until
1	1, 2, 3	Queues 2 and 3 are empty
2	1, 5	Both queues are empty
3	3, 4	Queue 4 is empty

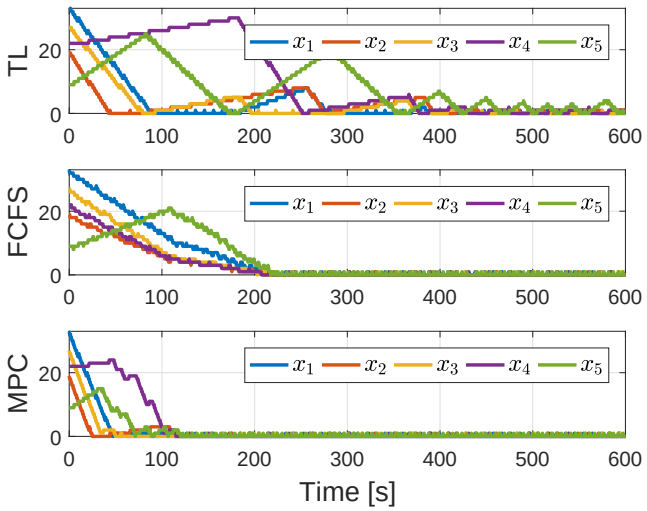

Fig. 4. The queue lengths over time for the five-lane intersection for the traffic light (TL), First-Come-First-Served (FCFS) and MPC controllers.

steps, and a sampling time of $\Delta t = 0.425$ seconds. To make a fair comparison, the other control methods use the same sampling time. The initial queue lengths are equal to $x(t_0, j_0) = [33\ 19\ 27\ 22\ 9]^\top$. The results are shown in Fig. 4. The average steady-state costs are 3.03 for human drivers with the traffic light, compared to 0.24 for the FCFS policy and 0.21 for MPC with automated vehicles.

Fig. 4 clearly shows the differences in the transient response among the three controllers. The traffic light goes through its cycle, emptying two or three queues at a time, and reaches steady-state after about 500 seconds. On the other hand, the FCFS policy constantly switches between the five queues, reaching steady-state after about 200 seconds. Finally, the MPC controller occasionally switches between queues 4 and 5, while keeping queues 1, 2 and 3 short, and reaches steady-state after only 100 seconds. It is clear from these results that MPC outperforms both the FCFS policy and the traffic light in its transient response. MPC similarly outperforms both the FCFS policy and the traffic light in steady-state. Additionally, it is clear that automating vehicles, which decreases the service times from (25) to (26), leads to a significant reduction in the average steady-state cost of a factor of 13–14.

Finally, the simulations, involving both the simulation of the queuing model and solving the optimization problem each sampling period, run faster than real-time, which indicates that the real-time implementation of the proposed approach is indeed feasible.

6. CONCLUSIONS

This paper presents a Model Predictive Control (MPC) approach to optimise the crossing sequence for the Cooperative Intersection Control methodology first presented in Morales Medina et al. (2015, 2018). First, the intersection is modelled as a hybrid system, which describes the arrival and departure of individual vehicles at queues at the boundary of the Cooperation Zone. Then, the controller is designed using MPC, which minimises the queue lengths and respects the service times between any two vehicles to guarantee safe passage through the intersection. Simulation results show that MPC achieves a faster transient response and a steady-state response with lower average costs than both an FCFS policy and a traffic light.

REFERENCES

Colombo, A. and Vecchio, D.D. (2015). Least restrictive supervisors for intersection collision avoidance: A scheduling approach. *IEEE Transactions on Automatic Control*, 60(6), 1515–1527.

Creemers, F. (2017). *Optimal Cooperative Intersection Control*. Master's thesis, Eindhoven University of Technology, the Netherlands.

Dresner, K. and Stone, P. (2008). A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31, 591–656.

Fleuren, S. and Lefeber, E. (2016). Data of real-life intersections for fixed-time traffic light control. URL `https://pure.tue.nl/ws/files/26651263/DataOfRealLifeIntersections.pdf`.

Gallestey, E. et al. (2003). Using model predictive control and hybrid systems for optimal scheduling of industrial processes. *Automatisierungstechnik*, 51(6), 285–293.

Goebel, R., Sanfelice, R.G., and Teel, A.R. (2009). Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2), 28–93.

Goebel, R., Sanfelice, R.G., and Teel, A.R. (2012). *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, 502–510.

Hult, R., Zanon, M., Gros, S., and Falcone, P. (2016). Primal decomposition of the optimal coordination of vehicles at traffic intersections. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2567–2573.

Morales Medina, A.I., van de Wouw, N., and Nijmeijer, H. (2015). Automation of a t-intersection using virtual platoons of cooperative autonomous vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, 1696–1701.

Morales Medina, A.I., van de Wouw, N., and Nijmeijer, H. (2018). Cooperative intersection control based on virtual platooning. *IEEE Transactions on Intelligent Transportation Systems*, 19(6), 1727–1740.

Qian, X., Gregoire, J., de La Fortelle, A., and Moutarde, F. (2015). Decentralized model predictive control for smooth coordination of automated vehicles at intersection. In *2015 European Control Conference*, 3452–3458.

Sanfelice, R.G., Copp, D.A., and Nanez, P. (2014). Hybrid Equations (HyEQ) Toolbox. URL `http://www.mathworks.com/matlabcentral/fileexchange/41372-hybrid-equations-toolbox-v2-01`.

Torrisi, F., Bemporad, A., and Mignone, D. (2000). HYSDEL — A Tool for Generating Hybrid Models. URL `http://people.ee.ethz.ch/~cohysys/hysdel/`.