Journal of Physics: Conference Series **659** (2015) 012049 doi:10.1088/1742-6596/659/1/012049

Stability of distributed MPC in an intersection scenario

T Sprodowski[1] **and J Pannek**[1,2]

[1] Department of Production Engineering, University of Bremen
[2] BIBA Bremer Institut für Produktion und Logistik GmbH, 28359 Bremen, Germany

E-mail: {spr[1], pan[2]}@biba.uni-bremen.de

Abstract. The research topic of autonomous cars and the communication among them has attained much attention in the last years and is developing quickly. Among others, this research area spans fields such as image recognition, mathematical control theory, communication networks, and sensor fusion. We consider an intersection scenario where we divide the shared road space in different cells. These cells form a grid. The cars are modelled as an autonomous multi-agent system based on the Distributed Model Predictive Control algorithm (DMPC). We prove that the overall system reaches stability using Optimal Control for each multi-agent and demonstrate that by numerical results.

Today the development of autonomous cars and the insurance of their correct behavior are still a great challenge in both academic research and the automotive industry. To ensure that networked cars are reliable in shared-space scenarios, the car-to-car communication implemented as a cooperative strategy is a possibility for getting a solution in the intersection resource problem. The ideal situation will be if all participants support them. Yet, non-communicating participants should also be included within such scenarios. Here, we assume that even if communication data is missing, a networked car is able to estimate the future route of a respective participant sufficiently accurate to guarantee collision avoidance.

To incorporate the behavior of the other cars and of disturbance factors, we propose to utilize a Distributed Nonlinear Model Predictive Control Scheme to formulate our problem as an optimal control problem. Nonlinear Model Predictive control (NMPC) is nowadays a widespread method, which is used in many feedback control system applications [1]. A survey of the nonlinear version is given by [2] and [3]. NMPC itself is a three step procedure: First, the current state of the system is estimated. Then, based on that estimate, an internal model of the system dynamics is used to predict the state trajectory over a finite time horizon and to compute a control u minimizing a given cost functional subject to possible constraints. Last, only the initial segment of the computed control u is applied, which renders the algorithm to be iteratively applicable and repeat the procedure until the equilibrium is achieved. The goal of NMPC is to (practically) asymptotically stabilize the system to a target x^*. To guarantee this property, endpoint constraints [4] or terminal constraints and costs [5] as well as a relaxed Lyapunov inequality [3] have been proposed. Since terminal constraints are inadequate for a networked car scenario due to the size of the problem and also limit the feasible set [6], we follow the latter approach.

For our networked car scenario, a distributed control setting appears naturally. Stemming

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

from a centralized idea, Richards and How [7], among others, proposed to split up the centralized optimal control problem into subproblems connected by a communication network, which lead to a Distributed Model Predictive Controller (DMPC). The general idea is to reduce the complexity of the problem induced by the number of constraints and control variables, which are needed by the centralized problem. While the total number of constraints and control variables remains unchanged, each subproblem is much smaller and can be solved more easily. We model these subproblems as an independent NMPC problems, and connect them to obtain all information about the chosen solution of the other subproblems. Here, we follow the approach to a fixed order, in which subproblems are solved.

The paper is organized as follows: In the following section, we describe the intersection scenario, which we consider throughout our work. In Section 2, we formally introduce the DMPC setting for our problem scenario. Respective stability conditions are shown to hold in Section 3 and illustrated using numerical experiments in Section 4. Last, we draw some conclusions in Section 5 and give a short outlook on future work.

1. Intersection Scenario

We consider a scenario based on an intersection without traffic lights similar to [8]. In this scenario, the authors proposed to divide the shared space of an intersection into multiple cells. These cells form a two-dimensional grid, which is administrated centrally. This scenario illustrates a simple form of a shared resource allocation problem. The cars are managed by one multi-agent each, which aims to cross the intersection in two directions straight ahead. The resulting problem is solved by a centralized approach utilizing the Organic Computing Principle [9]. An observer/controller-architecture is introduced to calculate a path through the intersection for each incoming networked car and to detect deviations using the A^* Search Algorithm [10]. The path is formulated as a sequence of tuples. Each tuple contains time (n) and space coordinates (k, m) to clarify when the cells will be occupied ($p = (n_1, k_1, m_1), \ldots (n_i, k_i, m_i)$). Here, n represents the nth discrete time step, k and m are the coordinates within the grid. Since the cars are modelled as multi-agents which can re-plan their original path, deviations are possible. The observer recognizes the latter and influences the affected cars including the causer by renewing the paths and spreading the information. The communication is centralized and controlled by the observer-/controller-system. Hence, only car-to-infrastructure communication is considered.

We propose to modify this scenario and implement a distributed control. Different from [11] where the paths are predefined and the optimization function considers speed and acceleration only, our aim is to model both the car and the scenario more generally. The collision avoidance in [11] is introduced as a constraint, which is imposed to a car following the priority assignment. We consider the reservation of grid cells to avoid any collisions. An illustration of our scenario is illustrated in Figure 1. To this end, we allow traffic from each direction within our extended

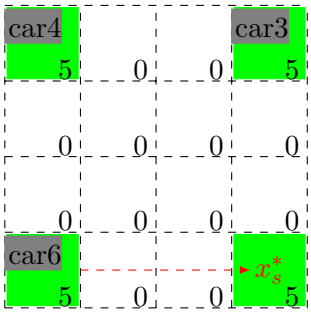

Figure 1. Example of a 4 × 4 - Intersection scenario divided in grid cells which shows the requested reservations made by cars, dashed line illustrates the planned path for car 6 to reach its target x_s^*

scenario. This raises the risk of a deadlock, which, for instance, may occur if two cars stand opposite to each other and cannot give way. Here, we avoid this possibility by introducing lanes, which are a necessity for conventional cars during the introduction phase of networked cars. The lanes determine the direction in which a car can move forward. Additionally, we introduce one multi-agent per car, which performs the path calculation via a DMPC algorithm to compute an optimal collision-free solution through the grid. To follow the approach of a fixed order, each car is set up with an index by initialization. The simulation procedure is executed as described in Algorithm 1.

Algorithm 1 Simulation procedure

Given: Initial time and set of active cars S.
While $S \neq \emptyset$ do

(i) **State estimation:**
Get current state for each car $x_s \in S$.

(ii) **Control computation:**
For each car $x_s \in S$ do

(a) If current position is target position, then remove car x_s from S and break.
Else set completed state to false.

(b) While completed state is false

1. Compute optimal path over finite horizon for given constraints.
2. Try to reserve time/space tuples. If successful, set completed state to true. Else, increment time value and try again.

(iii) **Control implementation:**
Implement all first control elements and shift horizon forward in time.

We model the target distance as a cost function and let the path be calculated and optimized by each vehicle separately. The actual path optimization in every time step n is necessary as we have multiple lanes for each direction so the dynamics is in the path calculation. We consider the reservation of grid cells to avoid any collisions.

2. Formulation in DMPC

2.1. System definition

We suppose the cars to be modeled as multi-agents and denote the overall state of the system by x, where there can be S subsystems

$$x = \{x_1, \cdots, x_S\}.$$

where s denotes the sth car. From the physical point of view x describes the position of the car in the intersection scenario. The path p through the intersection scenario is given as a sequence of tuples $(n, k, m) \in \mathbb{N}_0 \times \{1, \ldots, L_k\} \times \{1, \ldots, L_m\}$ with $L_k, L_m \in \mathbb{N}$, where the latter set denotes the time-space grid. The term (n, k, m) denotes the coordinates, at which a car can be situated, i.e.

$$x_s(n) = (n, k, m) \in \mathbb{N}_0 \times \{1, \ldots, L_k\} \times \{1, \ldots, L_m\}$$

where n denotes the discrete time, and k and m are the coordinates within the grid. The dynamics of the sth subsystem

$$x_s(n+1) = f_s(x_s(n), u_s(n)) \tag{1}$$

connects these tupels to generate a path, where the control $u_s(n)$ implements the path item for the next time step. Hence, the path can be written as

$$\text{Path } p_s = \left(x_s\left(n\right)\right)_{n\in\mathbb{N}_0}$$

The target for the overall system is denoted by x^* containing the targets

$$x^* = \left(x_s^*\right)_{s=\{1,\cdots,S\}}$$

for each subsystem, which we consider to be time invariant for simplicity of exposition. The control $u_s(n)$ of a subsystem s is given also as absolute coordinates of the grid similar to $x_s(n)$ which contains also the path tuple (n, k, m) for each system change.

2.2. Constraints of Subsystems

Each subsystem is contemplated with constraints that have to be satisfied. Here, we introduce three different kinds of constraints, the first to incorporate restrictions solely based on the local state and control, the second to include collision avoidance, and the third to enforce stability of each subsystem. The latter is modeled via

$$\#x_s = |\left(\left(n_1, k_1, m_1\right), \cdots, \left(n_i, k_i, m_i\right)\right)| < M, \tag{2}$$

where M defines an upper bound on the length of a path sequence. As a consequence, each subsystem has to reach its target in at most M steps.

To avoid collisions, we do not allow two subsystems x_r and x_v, $r \neq v$, to take identical time-space coordinates. Hence, a tuple (n_i, k_i, m_i) can only appear at most once in all paths p_s. More formally, we obtain

$$\forall \left(n_i, k_i, m_i\right) \in x_r \quad \Rightarrow \quad \left(n_i, k_i, m_i\right) \notin x_v \tag{3}$$

for all $r, v \in \{1, \cdots, S\}$ with $r \neq v$.

Last, using local restrictions we restrict the motion of each car by either staying at the current position, i.e. $x_s\left(n+1\right) = x_s\left(n\right)$, or moving to a neighbouring cell, that is from $x_s\left(n\right) = \left(n, k_1, m_1\right)$ to $x_s\left(n+1\right) = \left(n+1, k_2, m_2\right)$, where

$$\left(|k_2 - k_1|, |m_2 - m_1|\right) \leq (1, 1) \tag{4}$$

2.3. Costs for each Subsystem

Incorporating the dynamics (1) and the constraints (2), (3) and (4), each multi-agent optimizes its subsystem with respect to the stage cost

$$\ell\left(x_s(k), u_s(k)\right) = \|x_s(k) - x_s^*\|^2 + \lambda\|u_s(k)\|^2,$$

where $\lambda > 0$ is a regularization factor to avoid bang-bang behavior of optimal solutions. Utilizing the DMPC approach, we define the cost function for each subsystem via

$$J_s^N\left(x_s^0, u_s\right) := \sum_{k=0}^{N-1} \ell_s\left(x_s\left(k\right), u_s\left(k\right)\right) \tag{5}$$

where x_s^0 denotes the initial value of the path to be optimized, i.e. $x_s(0) := x_s^0$, and J_s^N the cumulated costs for the each system state k over an horizon of length N.

2.4. Costs of Overall System
The cost function can also be interpreted as energy, i.e. the overall system loose potential energy if the cars are moving towards the target. Here, we assume that after each time step of the DMPC procedure, the overall system looses potential energy. While appearing restrictive, this assumption always holds due to our fixed order of computations for subsystems. Hence, at least one car x_s can move forward and reduces its own costs, and thereby also the cumulative system costs. The DMPC procedure implements a minimizer of (5), which we denote by

$$u_s^*(\cdot, x_s) = \underset{u_s}{\operatorname{argmin}} J_s^N(x_s, u_s).$$

Hence, the cumulative costs of the overall system for the DMPC setting are given by

$$V^N(x^0) = \sum_{i=1}^{S} J_s^N\left(x_i^0, u_s^*(\cdot, x_i^0)\right) \tag{6}$$

2.5. Feedback of the subsystem
Last, we obtain the feedback μ_s^N for each subsystem as the first element of the minimizer $u_s^*(\cdot, x_s)$ from (6), i.e.

$$\mu_s^N(x_s^0) = u_s^*(0, x_s^0) \tag{7}$$

Note that the feedback is implicitly coupled with the success of reserving a path item (n, k, m) of the grid in the optimization and coordination step. If more than one system attempt to get a reservation for the same cell at the same time n, only the highest priority path item is accepted. This fulfills the constraint formulated in (3). The other inquires are rejected and the systems have to attempt a new reservation at a later time or consider a different field to attempt a reservation.

3. Stability conditions

Our goal is to show asymptotic stability of the overall system. We use [3, Def. 2.14] to derive an upper bound for our system, which will guarantee the asymptotic stability of the system. To define the required bounding function $\beta(r, n) \in \mathcal{KL}$, we introduce the set $\mathcal{K}$ describing the functions $\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, which are strictly increasing and satisfy $\alpha(0) = 0$. If additionally α is unbounded, then it called a class $\mathcal{K}_\infty$ function. Secondly, we call functions $\delta : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ to be of class $\mathcal{L}$, if δ is continuous and strictly decreasing with $\lim_{n\to\infty} \delta(t) = 0$. Combining these properties, we call $\beta(r, n)$ to be a class $\mathcal{KL}$ function if $\beta(r, n)$ is continuous and $\beta(\cdot, n) \in \mathcal{K}$ for all $n \geq 0$ and $\beta(r, \cdot) \in \mathcal{L}$ for all $r \geq 0$. Based on the asymptotically stability we can show that our system is optimal controllable. This is presented in the numerical results 4 where the suboptimality α is used to evaluate the performance of the controller.

Note that for our intersection scenario, we cannot guarantee that a minimizer u_s^* to exist such that costs are decreasing for each car in each time step. Indeed, the constraints (2) and (3) allow cars to hold their position for a number of time steps to let other cars pass. Yet, some cars move closer to their target and the respective costs of these subsystems are decreasing. Hence, only an overall consideration is promising.

In the following, we prove stability of the presented intersection scenario by showing that for $r := |x_s - x_s^*|$

$$\beta(r, n) = S \max\left\{\overline{M} - \underline{M} - n, 0\right\}\underline{M}^2 + \lambda S \underline{M} + \sum_{j=1}^{\underline{M} - \max\left\{0, n - \overline{M} + \underline{M}\right\}} j^2$$

is a class $\mathcal{KL}$ function and by constructing an upper bound for the overall costs. To this end, we first derive a bound $\beta(r,0)$ for the initial conditions, which we extend in a second step to an upper bound for all time instances n. Here, we abbreviate the lower bound of the path length (2) as $\underline{M} = \min(\#x_s)$ and the upper bound as $\overline{M} = \max(\#x_s)$.

First, we ignore (3) and define $\beta(r,0)$ as a function, where all S cars stay in their initial position until they have to leave to get to the target without violating $\overline{M}$. Due to ignoring the collision constraint (3), $\beta(r,0)$ represent an upper bound on the costs. We split this expression into two subsums:

$$\beta(r,0) = \sum_{s=1}^{S} \left(\sum_{k=0}^{\overline{M}-\underline{M}-1} \ell(x_s,0) + \sum_{k=\overline{M}-\underline{M}}^{\overline{M}-1} \ell\left(x_s(k)\right), u_s(k) \right).$$

Since by assumption cars remain at their initial position, we have $u_s(k) = 0, k = 0, \cdots, \overline{M} - \underline{M} - 1$ and can dissolve the inner sums to

$$\begin{aligned}\beta(r,0) &= \sum_{s=1}^{S} \left(\left(\overline{M} - \underline{M}\right) ||x_s^0 - x_s^*||^2 + \sum_{k=\overline{M}-\underline{M}}^{\overline{M}-1} ||x_s(k) - x_s^*||^2 + \lambda ||u_s||^2 \right) \\ &= \sum_{s=1}^{S} \left(\left(\overline{M} - \underline{M}\right) ||x_s^0 - x_s^*||^2 + \sum_{k=0}^{\underline{M}-1} ||x_s(k) - x_s^*||^2 + \lambda ||u_s||^2 \right),\end{aligned}$$

which allows us to shift the interval for the second sum to $k = 0, \cdots, \underline{M} - 1$. By our model definition, we obtain $||x_s^0 - x_s^*|| = \underline{M}$ for the initial position and $||x_s(k) - x_s^*|| = \underline{M} - k$ as a car moves towards its target with $||u_s||^2 = 1$. Hence, we obtain

$$\beta(r,0) = \sum_{s=1}^{S} \left(\left(\overline{M} - \underline{M}\right) \underline{M}^2 + \sum_{k=0}^{\underline{M}-1} \left(\underline{M} - k\right)^2 + \lambda \underline{M} \right).$$

Dissolving now the inner sum over the systems we get

$$\beta(r,0) = S \left(-\underline{M}^3 + \underline{M}^2 \overline{M} + \underline{M}\lambda + \sum_{k=1}^{\underline{M}} k^2 \right). \tag{8}$$

With the application of square pyramidal number for the last remaining sum in (8)

$$\sum_{k=1}^{\underline{M}} k^2 = \frac{1}{6} \left(\underline{M} + 1\right) \underline{M} \left(2\underline{M} + 1\right)$$

we can simplify (8) to

$$\beta(r,0) = S \left(-\frac{2}{3}\underline{M}^3 + \underline{M}^2 \left(\overline{M} + \frac{1}{2}\right) + \underline{M} \left(\lambda + \frac{1}{6}\right) \right), \tag{9}$$

which reveals an upper bound on the initial cost. To construct the decrease of the bound over time, we consider the sum over the subsystems S

$$\beta(r,0) = \sum_{s=1}^{S} \left(\frac{1}{3} \|x_s(k) - x_s^*\|^3 + \|x_s(k) - x_s^*\|^2 \left(k + \frac{1}{2}\right) + \|x_s(k) - x_s^*\| \left(\lambda + \frac{1}{6}\right) \right).$$

Now, we can use (9) to incorporate the time n in $\beta(r,n)$. The upper bound over time resembles the sequential steps of all cars. Hence, as time progresses, we get

$$\beta(r,n) = \sum_{s=1}^{S} \left(\sum_{k=0}^{\max\{\overline{M}-1-n,0\}} \ell(x_s(0),0) + \sum_{k=\max\{\overline{M}-\underline{M}-n,0\}}^{\max\{\overline{M}-1-n,0\}} \ell(x_s(k+n),u_s(k+n)) \right).$$

Again, the first inner sum describes those cars waiting at their position until they must start to reach their target. The second sum represents the costs over the interval during which the cars move towards their target. Dissolving the first sum reveals

$$\beta(r,n) = S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 + S \sum_{k=\max\{\overline{M}-\underline{M}-n,0\}}^{\max\{\overline{M}-1-n,0\}} \ell(x_s(k),u_s(k)).$$

By shifting the index of the remaining sum to $k=0$, we get

$$\begin{aligned}\beta(r,n) = {} & S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 \\ & + S \sum_{k=0}^{\substack{\max\{\overline{M}-1-n,0\} \\ -\max\{\overline{M}-\underline{M}-n,0\}}} \ell\left(x_s\left(k+\max\{\overline{M}-\underline{M}-n,0\}\right),u_s\left(k+\max\{\overline{M}-\underline{M}-n,0\}\right)\right).\end{aligned}$$

As we combine the difference of the maxima of the upper limit, we get

$$\begin{aligned}\beta(r,n) = {} & S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 \\ & + S \sum_{k=0}^{\underline{M}-1} \ell\left(x_s\left(k+\max\{\overline{M}-\underline{M}-n,0\}\right),u_s\left(k+\max\{\overline{M}-\underline{M}-n,0\}\right)\right).\end{aligned}$$

Resetting $k := k-n$, the latter expression simplifies to

$$\begin{aligned}\beta(r,n) = {} & S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 \\ & + S \sum_{k=0}^{\underline{M}-1} \ell\left(x_s\left(k-n+\max\{\overline{M}-\underline{M},n\}\right),u_s\left(k-n+\max\{\overline{M}-\underline{M},n\}\right)\right).\end{aligned}$$

As $\overline{M}-\underline{M} \geq n$, we can eliminate the maximum terms in the index and get therefore

$$\begin{aligned}\beta(r,n) = {} & S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 \\ & + S \sum_{k=0}^{\underline{M}-1} \ell\left(x_s\left(k-n+\overline{M}-\underline{M}\right),u_s\left(k-n+\overline{M}-\underline{M}\right)\right). \qquad (10)\end{aligned}$$

and for $\overline{M}-\underline{M} \leq n$

$$\beta(r,n) = S \max\{\overline{M}-\underline{M}-n,0\}\,\underline{M}^2 + S \sum_{k=0}^{\underline{M}-1} \ell(x_s(k),u_s(k)). \qquad (11)$$

as distinction of both cases.

As $k+\overline{M}-\underline{M}-n \leq k+\overline{M}-\underline{M} \leq k+\overline{M}$ for case (10) we obtain

$$\beta(r,n) = S \max\left\{\overline{M}-\underline{M}-n,0\right\}\underline{M}^2 + S\sum_{k=0}^{\underline{M}-1}\left(k+\overline{M}\right)^2 + \lambda S1$$

and for the second case (11)

$$\beta(r,n) = S \max\left\{\overline{M}-\underline{M}-n,0\right\}\underline{M}^2 + S\sum_{k=0}^{\underline{M}-1} k^2 + \lambda S1$$

For the first case (10) we can assume that for the index $k+\overline{M}-\underline{M}-n$ the inequality $k \leq n-\overline{M}+\underline{M}$ and for the second case (11) $k \leq \underline{M}-1$ as given by the interval.

Therefore we can dissolve both cases by substitute with $j := \underline{M}+k$ and $\max\left\{0, n-\overline{M}+\underline{M}\right\} = \underline{M}-j$ to get $\underline{M}-1 = \underline{M}-j$ which leads to $j=1$ for the lower bound and we get

$$\beta(r,n) = S \max\left\{\overline{M}-\underline{M}-n,0\right\}\underline{M}^2 + \lambda S\underline{M} + \sum_{j=1}^{\underline{M}-\max\left\{0,n-\overline{M}+\underline{M}\right\}} j^2$$

Hence, by construction we have that $\beta(r,n) \in \mathcal{KL}$ and an upper bound of the overall costs satisfying the conditions of [3, Def. 2.14]. Utilizing the properties of the system and of $\underline{M}$, $\overline{M}$, the function $\beta(r,n)$ may largely overestimate the costs of the overall system. These properties allowed us to exclude (3) and avoid a collision detection as required in the closed loop. Thereby, we can conclude asymptotic stability of the overall closed loop system.

Within the analysis of the constructed feedback law, we are not only interested in stability of the closed loop, but also in the performance of the control. Since the bound $\beta(r,n)$ is a worst case bound, the performance estimate that can be derived from it is rather poor. Additionally, the values $\underline{M}$, $\overline{M}$ may be unknown for example in that case as we cannot predict the maximum wait time of one car which can prolong the path length. To circumvent these issues, we make use of the concept of relaxed Lyapunov arguments in the trajectory based setting. In particular, [3, Proposition 7.6] shows that if there exist a constant $\alpha \in (0,1]$ such that the relaxed Lyapunov inequality

$$V^N(x(n)) \geq \alpha \ell\left(x(n), \mu^N(x(n))\right) + V^N(x(n+1)) \tag{12}$$

holds for all $n \in \mathbb{N}_0$ and additionally there exist functions $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$ such that

$$\alpha_1(\|x(n)\|) \leq V^N(x(n)) \leq \alpha_2(\|x(n)\|), \tag{13}$$

$$\alpha_3(\|x(n)\|) \leq \ell\left(x(n), \mu^N(x(n))\right), \tag{14}$$

then the closed loop solution $x(n)$ behaves like a trajectory of an asymptotically stable system and the performance bound

$$J^\infty\left(x(0), \mu^N(x_s(0))\right) \leq V^N(x(0))/\alpha \leq V^\infty(x(0))/\alpha$$

holds.

Since we have shown the existence of a bound $\beta(r,n)$, we can apply converse Lyapunov results to guarantee the existence of α_1, α_2, $\alpha_3 \in \mathcal{K}_\infty$ such that (13), (14) hold, cf. [12]. This allows us to concentrate on the performance evaluation of the proposed control scheme via (12).

4. Experiments

For the implementation we use the discrete event simulator framework adevs [13] along with the Qt framework [14]. The cars are modelled as multi-agents objects, which include a controller instance, dynamical properties, constraints and a cost function with individual targets. The grid is implemented as an array containing all grid cells. Upon initialization, each car reserves a time and grid cell tuple in first-come-first-serve fashion. Hence, if a car enters the scenario at a later time instant, its initial time will also be later. To retain a unique order for the reservation process, the cars are indexed. For the optimization step within the MPC, we apply the COBYLA Algorithm (**C**onstrained **O**ptimization **BY** **L**inear **A**pproximations) developed by Powell [15] provided by the NLOpt library [16]. Minimizing the cost function including the individual target of the car implicitly defines the direction and route of the car. The model of the middle lane is given as a set of constraints which are defined for each car depending on its start position. Both the path and control values x_s, u_s are mapped from tuples to absolute values of the cell grid. In particular, we map the cell coordinate tuples (n, k, m) to global scalars w via

$$\vec{w} = (w_1, \ldots, w_N), w_i \in \{0, \ldots, L_k L_m - 1\}$$

with

$$w_i = k_i + m_i L_k.$$

If a car reaches its target, the car is removed from the scenario and is not considered in further calculations. We set up a standard intersection scenario with a 4×4-intersection grid and 20 cars starting in the corners. To guarantee a connected path, we imposed a penalty term η. Here, a path is connected if each pair of consecutive path items are neighbors in the grid. The penalty is defined as

$$\eta_s = Bv,$$

where v is the number of not connected pairs of consecutive path items and B a constant.

To analyze the performance of the controller, we utilize the minimal and maximal path lengths

$$p_{\min} = \min_s \#x_s \geq \underline{M}, \quad p_{\max} = \max_s \#x_s \leq \overline{M},$$

the mean path length

$$\bar{p} = \frac{1}{S} \sum_{s=0}^{S} \#x_s.$$

and the suboptimality bound α from (12). We evaluate the bound α as the minimum over all time steps $n \in \mathbb{N}_0$ and compute the mean over all systems $x_1, \ldots, x_S$ via

$$\alpha_n = \frac{\sum_{i=1}^{S} \min\left(\alpha_{n-1}, \frac{V_{n-1} - V_n}{\ell_{n-1}}\right)}{S} \quad \text{with } \alpha_{-1} = 1.$$

For α we used the value in the last step named here α_n and $\overline{\alpha}$ as the mean over all time steps.

For our experiments, we considered the parameters $\lambda = 0.2$ and $B = 100$ and a horizon length of $N = 3$.

We set up configurations with different quantities of cars and the intersection size. Pursuing a realistic approach we consider a 4×4-intersection scenario, i.e. two lanes for each direction,

Table 1. Experimental results for test scenarios, α_n as last time step, $\overline{\alpha}$ as mean value over all time steps in the simulation

$\#S$	Grid	N	$p_{\min}$	$p_{\max}$	$\bar{p}$	α_n	$\overline{\alpha}$
20	4×4	3	4	11	6.65	0.25	0.2155
30	4×4	3	4	10	6.2	0.27	0.2707
20	6×6	3	5	18	7.25	0.0019	0.1005
30	6×6	3	5	13	6.76	0.00013	0.1639
20	6×6	4	5	11	6.55	0.00587	0.43
30	6×6	4	5	16	6.8	0.00588	0.3587

and a 6×6-intersection scenario reflecting 3 lanes for each direction. In Table 1, we present our numerical results for both scenarios with 20 and 30 cars for each direction.

In the 4×4-intersection we observe a slightly larger mean path length $\bar{p}$ for 30 cars than for 20 cars. This due to higher reservation times for the direct way from the start to the target cell of a vehicle as more cars try to reserve the direct way. With the growing reservation time values for the direct lane, the minimum of cost function is attained using the second lane. Hence, cars entering the intersection at a later time step choose the second lane. Reflecting this situation we can argue that a certain amount of cars is necessary that the cost function gives an advantage over distance for a shorter time delay. For α we used the value in the last step named here mentioned α_n and $\overline{\alpha}$ as the mean over all time steps.

For the 6×6-scenario, we similarly observe that the mean path length $\bar{p}$ is smaller for the configuration of 30 cars than with 20 cars. The effect is also increased with the greater intersection dimensions. Hence, we can conclude that the influence of the higher distances for the cars leads to the greater gap in $\bar{p}$. Additionally, we considered different horizon lengths and observed an decrease in $\bar{p}$ for larger horizon for 20 cars. For the scenario with 30 cars $\bar{p}$ changes only slightly. Again, the reservation of more cells in the optimization has a large impact on the difference of $\bar{p}$ for 20 cars with greater horizon length. The time slots which are reserved by the cars reveal later times for late arriving cars. Hence, the cost function leads to an earlier change to the second lane. Yet, we also observed an improvement in the suboptimality index for both scenarios which improves greatly for the larger horizon length.

In the following figures, we give some graphs for the overall cost function according to Table 1. We included $\beta(r, n)$ as a monotonously decreasing function representing the upper bound for our cost function. For longer horizons illustrated in Figure 6 and 7, we observe that the cost function tends to zero more slowly. In this case, the control impact is less aggressive compared to a shorter horizon, which is a known typical effect of using MPC. In the figures 4-7 a slightly increase of the costs occurs in one step. As mentioned in the 4x4-evaluation, this is caused by changing to the second lane of the later following cars. As the cost function returns lower costs due to earlier available reservation times for the second lane over the optimization horizon for one car than reserving later reservation times in the crowded cells, the applied optimal control $u(0)$ increases the costs for the changing cars slightly. Still, the costs are decreasing and remain bounded by $\beta(r, n)$, which indicates that the solutions show stable behavior.

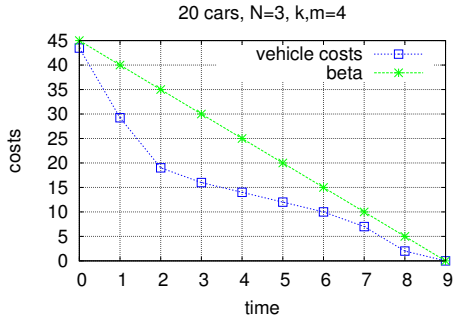

Figure 2. Overall costs for 20 cars, N=3, k,m=4

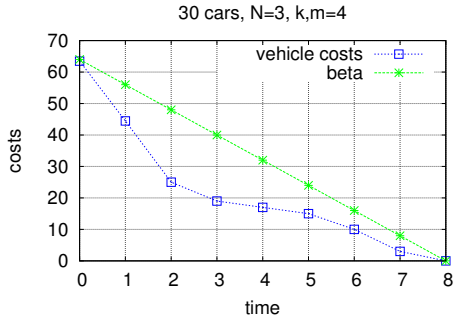

Figure 3. Overall costs for 30 cars, N=3, k,m=4

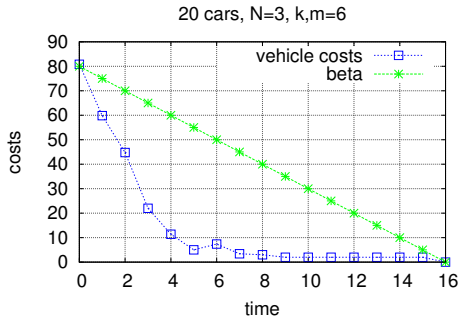

Figure 4. Overall costs for 20 cars, N=3, k,m=6

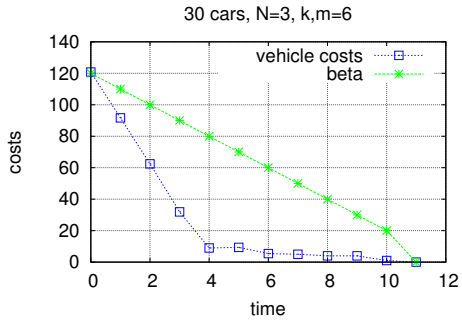

Figure 5. Overall costs for 30 cars, N=3, k,m=6

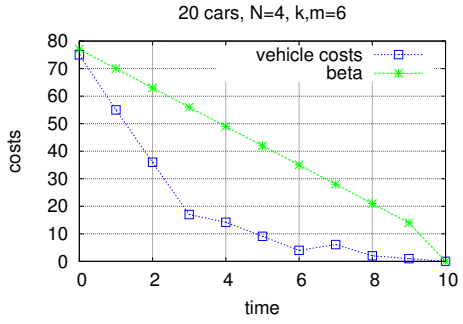

Figure 6. Overall costs for 20 cars, N=4, k,m=6

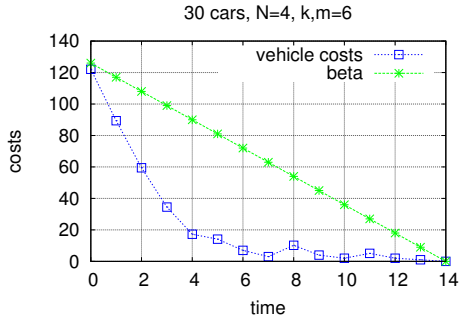

Figure 7. Overall costs for 30 cars, N=4, k,m=6

5. Conclusion

In this paper we used an intersection scenario as a discrete shared space grid. We formulated this scenario as a Distributed Model Predictive Control Problem in Section 2 where each car

modelled as an multi-agent optimizes its own path. In Section 3 we have proven the stability for the overall system by deriving a class $\mathcal{KL}$ boundary function. Our numerical results showed that the suboptimality index α known from relaxed Lyapunov theory remains positive. Additionally, the cumulated costs are decreasing in every time step and satisfy our computed bound, which indicates stable behavior of the system.

In further research we will consider an exponential term depending on time to privilege long waiting cars. Converting the cost function into a continuous one will also extend the suitability for continuous optimization algorithms. Additionally, we plan to incorporate the risk of deadlocks by allowing any direction for the lanes to increase the throughput for the overall traffic. Last, by including path predictions of non-communicating participants we will improve practicability of the coordination.

Acknowledgment
We thank Prof. C. Müller-Schloer from the Leibniz University of Hannover for a fruitful discussion on this topic.

References

[1] Qin S J and Badgwell T A 2003 A survey of industrial model predictive control technology *Control Engineering Practice* **11** 733–764
[2] Rawlings J B and Mayne D Q 2009 *Model Predictive Control: Theory and Design* (Nob Hill Pub.)
[3] Grüne L and Pannek J 2011 *Nonlinear Model Predictive Control: Theory and Algorithms* Communications and Control Engineering (London: Springer)
[4] Keerthi S S and Gilbert E G 1988 Optimal Infinite Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability and Moving Horizon Approximations *Journal of Optimization Theory and Applications* **57** 265–293
[5] Chen H and Allgöwer F 1998 A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability *Automatica* **34** 1205–1217
[6] Grüne L 2012 NMPC without terminal constraints *IFAC Proceedings Volumes (IFAC-PapersOnline)* **4** 1–13
[7] Richards A and How J 2003 Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility *Proceedings of the 2003 American Control Conference, 2003.* **5** 4034–40
[8] Chaaban Y, Hähner J and Müller-Schloer C 2009 Towards Fault-Tolerant Robust Self-Organizing Multi-agent Systems in Intersections without Traffic Lights *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns* 467–475
[9] Richter U and Mnif M 2006 Towards a generic observer / controller architecture for Organic Computing *GI Jahrestagung* **1** 112–119
[10] Hart P E, Nilsson N J and Raphael B 1968 A Formal Basis for the Heuristic Determination of Minimum Cost Paths *IEEE Transactions of systems science and cybernetics* 100–107
[11] Makarem L and Gillet D 2013 Model predictive coordination of autonomous vehicles crossing intersections *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* 1799–04
[12] Jiang Z P and Wang Y 2002 A converse Lyapunov theorem for discrete-time systems with disturbances *Systems and Control Letters* **45** 49–58
[13] Nutaro J 2013 adevs (A Discrete Event Simulator) library http://web.ornl.gov/~1qn/adevs/
[14] Digia Qt Framework
[15] Powell M J D 1998 Direct search algorithms for optimization calculations *Acta Numerica* **7** 287–336
[16] Johnson S The {NLopt} nonlinear-optimization package