A Piecewise Linearization Algorithm for Solving MINLP in Intersection Management

Matthias Gerdts[1][a], Sergejs Rogovs[1] and Giammarco Valenti[2]
[1]Institute of Engineering Mathematics, Bundeswehr University Munich, Germany
[2]Department of Industrial Engineering, University of Trento, Italy

Keywords: Connected and Automated Vehicles, Cooperative Driving, Intersection Management, Piecewise Linearization, Mixed Integer Nonlinear Programming.

Abstract: In this paper, we propose a linearization algorithm for solving a Mixed Integer NonLinear Problem (MINLP) for Intersection Management (IM) of Connected Autonomous Vehicles (CAVs). The objective of such problem is to minimize the time it takes to clear a given arbitrary intersection for all vehicles in the consideration. We treat the IM problem as a bi-level optimization problem. On the lower level we solve an Optimal Control Problem (OCP) for each individual vehicle, whereas on the higher level we deal with an optimization problem of finding the optimal sequence and starting times for every car, which essentially yields a MINLP. An intuitive linearization technique is presented to solve the emerging MINLP in a reasonable time. The actual controls, if necessary, are computed *a posteriori* by minimizing the L^2-norm of control variables. The algorithm is tested in different intersection scenarios. Numerical results show that it is suitable for real-time applications.

1 INTRODUCTION

Road intersections play a very important role in transportation networks due to safety and efficiency reasons, especially in urban areas. The rise of automation in the automotive field, combined with communication paradigms such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I), gave birth to new ways of addressing the problem. The concept of Connected Autonomous Vehicles allows to exploit the capabilities of self-driving vehicles. Optimal coordination of CAVs at intersections can help a lot in the improvement of traffic efficiency and safety aspects. Research on this topic gained a lot of interest in the last decade with a significant boost in the last three years, see e.g. (Namazi et al., 2019) or (Khayatian et al., 2020). Therein, one can find a detailed review of the literature on the topic. Since there exists a vast variety of algorithms on this topic, there are several ways how one can group them. For instance, the approaches can be grouped by optimization goals. The most common goals are: *safety, efficiency, ecology* and *passenger comfort*. Another way how to cluster the approaches is by their cooperation paradigm: *distributed* and *centralized*. In the distributed case, CAVs try to find an optimal crossing sequence by communicating with each other, whereas in the centralized case there is some managing unit located in the vicinity of the intersection. A examples of distributed approaches can be found in (Britzelmeier and Dreves, 2020), (Malikopoulos et al., 2018) and (Ahmane et al., 2013). In the first reference, the problem is represented as a generalized Nash equilibrium problem which results from a coupled optimal control problem. Moreover, to the best of our knowledge this is the first work on intersection scheduling with a proof of convergence.

In the underlying work, we consider a centralized approach. If no traffic priorities are given, the intersection management problem in this case can be addressed as a bi-level optimization problem. On the lower level the motion of each individual vehicle has to be found in accordance to some optimization goal, whereas on the upper level one tries to find an optimal crossing order such that no collisions occur, and some global cost is optimized. Often, this global cost is simply the total time elapsed to clear an intersection for all vehicles in consideration. Let us first mention some existing works with similar modeling/solution techniques, and then discuss the contribution of our work. The approaches given in (Fayazi and Vahidi, 2018), (Mohamad Nor and Namerikawa, 2019) exploit the same vehicle model and the same modeling

[a] https://orcid.org/0000-0001-8674-5764

Gerdts, M., Rogovs, S. and Valenti, G.
A Piecewise Linearization Algorithm for Solving MINLP in Intersection Management.
DOI: 10.5220/0010437104380445
In *Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, pages 438-445
ISBN: 978-989-758-513-5

techniques for scheduling constraint. The geometry of the intersection is instead restricted to a specific use case with no turns, and due to some simplifications, the upper level problem is modeled as a MILP. In (Hult et al., 2018), (Hult et al., 2019) a more sophisticated car model is considered which results in a MINLP on the upper level. The emerging MINLP is then solved using an SQP-type algorithm. See also (Hult et al., 2020) for a real-world experiment.

In this paper we consider a simple car model as in (Fayazi and Vahidi, 2018), (Mohamad Nor and Namerikawa, 2019), however, on the upper level we end up with a MINLP as in (Hult et al., 2018), (Hult et al., 2019), and in contrast to Hult et al., we solve the MINLP with some linearization technique inspired by (Winston and Goldberg, 2004). Such a choice is explained by existence of high-performance software packages for solving MILP. We show that our technique can be used for real-time applications.

The rest of the paper is organized as follows. In Section 2 we shape the scope of our considerations by discussing restrictions and assumptions we make regarding vehicles and intersection geometry. Section 3 is devoted to the problem formulation, where we introduce the vehicle model, define scheduling constraints, and state the final MINLP. Moreover, in this section we discuss the case when a car cannot enter an intersection at an optimal (in local sense) velocity. In Section 4 a piecewise linearization technique is introduced and a final MILP is stated. Finally, in Sections 5 and 6 we present some numerical experiments and state conclusions.

2 GENERAL CONSIDERATIONS AND ASSUMPTIONS

In this section we define the scope of our considerations. Here, we discuss all the restrictions we make related to vehicles and intersections in our consideration. Moreover, we summarize the most important quantities and variables we are using in Table 1.

2.1 Vehicles

For the sake of shortness we restrict our considerations to only one vehicle in each lane. This assumption seems to be a quite strict one, however the approach presented in this work can be easily generalized to an arbitrary number of vehicles in each lane by adding some constraints to guarantee that no rear-end collisions take place, see e.g. (Hult et al., 2019). Such a general setting is going to be considered in our future work. Throughout this paper, we denote by N the total number of vehicles and the corresponding index set by $\mathcal{V} = \{1, \dots, N\}$. Moreover, we assume that the states of each vehicle are known without any uncertainties. The states can be obtained by solving a simple optimal control problem for each vehicle, see Section 3.1.

Assumption 1. *There is at most one car in each lane.*

Assumption 2. *The states of each vehicle are known without any uncertainties.*

The last assumption related to vehicles we make, states that we deal only with feasible initial conditions. Such an assumption is straightforward. The upcoming MINLP problem cannot be solved for any set of initial conditions. The initial conditions that lead to an infeasible problem (impossibility to respect all constraints) are not considered. This assumption does not imply any loss of generality, since infeasible initial conditions eventually lead to an accident and they must be *a priori* excluded.

Assumption 3. *Only feasible initial conditions are considered.*

2.2 Intersection Geometry

Now, let us define some assumptions on the intersection geometry. The approach presented in this work is conceived to be flexible and can be easily adapted to any arbitrary intersection.

Lanes: The lanes that *enter/exit* an intersection are called *entering/exit* lanes. It is not allowed to overtake within the same lane. Moreover, no lane change is allowed (vehicle is supposed to be already in the correct lane) in the portion of the intersection considered in the problem. These assumptions are widely used for such problems (Namazi et al., 2019).

Assumption 4. *No overtake or lane change maneuvers are allowed.*

Paths: Every entering lane can be connected to each exit lane with some predefined path. The number of all paths is denoted by M and the corresponding index set reads as $\mathcal{P} = \{1, \dots, M\}$. The corresponding vehicle-to-path mapping reads

$$P : \mathcal{V} \to \mathcal{P},$$

where for every $i \in \mathcal{V}$ there exist exactly one $k \in \mathcal{P}$ such that $P(i) = k$. In other words, to cross an intersection each car follows exactly one path which depends on the car's route. Each path's trajectory is assumed to be explicitly known. See an example of intersection geometry in Fig. 1.

Assumption 5. *Each vehicle follows exactly one predefined path.*

Conflict Zone: A *conflict zone* (CZ) is the area of an intersection that contains all the overlapping points of the all paths. All the possible trajectory collisions can take place only in this area. For simplicity it is usually assumed that a CZ is given by some polygon, see Fig. 1 where a canonical intersection is considered, whose CZ is given by a square.

Once a conflict zone is defined, the longitudinal motion of each vehicle is broken down into two phases. Namely, the *approaching phase* (fully described in Section 3.1) which deals with the vehicle's motion from the initial position to the point where the conflict zone begins, and the *intersection phase* which starts where the first phase finishes and ends when a vehicle leaves the conflict zone. For simplicity reasons the velocity of each vehicle is assumed to be constant during the intersection phase. This assumption is also widely used in the literature, see e.g. (Namazi et al., 2019).

Assumption 6. *The velocity of each vehicle during the intersection phase is constant.*

Conflict Matrix: In order to capture all possible combinations of pairwise intersecting paths, which is essential for collision avoidance reasons, a so-called *conflict matrix* is introduced. Entries of such conflict matrix can take only binary values. We define the conflict matrix as follows. For all pairs $k,l \in \mathcal{P}$, the corresponding entry of the conflict matrix is given by

$$c_{kl} = \begin{cases} 0, & \textit{paths do not overlap,} \\ 1, & \textit{paths overlap, there is conflict.} \end{cases} \quad (1)$$

Such a matrix is essential for the formulation of scheduling constraints (7). Note that in order to avoid duplicate constraints, by convention, the values of the lower triangle of the matrix must be set to zero.

In a general case, where we have more than one car in a lane, a conflict matrix will possess a more complex structure. However, as already mentioned above, such complex cases are not considered in the underlying work and will be addressed in one of our future publications.

3 PROBLEM FORMULATION

This section is the modeling part of the underlying work. Here, we discuss the tools and techniques we need to solve the intersection scheduling problem. Let us briefly discuss the steps we take in this section. First, we introduce a simple vehicle model and N optimal control problems, which maximize the velocity of each car at the end of the approaching phase. The choice of such objective function is due to Assumption 6. These simple OCPs can be solved analytically by treating the final times (of corresponding approaching phases) as parameters. The results of this problems are final velocities (6) as functions of final times. Next, we define scheduling constraints (7) which are essential for collision avoidance in the CZ, and state mixed integer nonlinear problem (10), which gives us an optimal intersection entering sequence in terms of total time elapsed. Finally, we post-regularize the motion of those vehicles that are subject to velocity saturation.

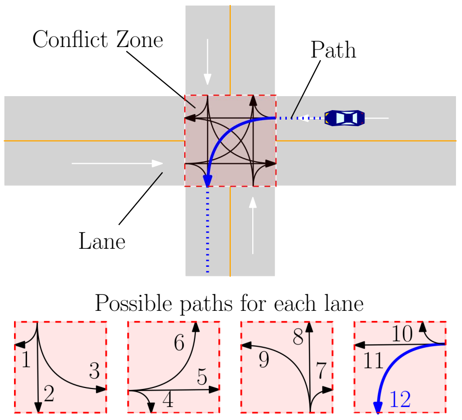

Figure 1: Sample of an intersection geometry. In the upper part of the figure a classical road intersection is shown. It consists of 4 entering lanes, 4 exit lanes and 12 possible paths that are shown in the lower part of the figure. The path number of the sample vehicle 1 is 12, which corresponds to $P(1) = 12$.

3.1 Vehicle Model and Motion

For simplicity reasons the selected vehicle model is a double integrator. The states of each vehicle are given by its curvilinear coordinates along a predefined path $s_i(t),\ i \in \mathcal{V}$, and its derivatives (longitudinal velocities) $v_i(t),\ i \in \mathcal{V}$. The coordinates $s_i(t)$ have to vanish at the end of the approaching phase (at the beginning of the intersection phase). The velocity is assumed to be non–negative and bounded from above. The controls are simply longitudinal accelerations denoted by $a_i(t),\ i \in \mathcal{V}$, which are bounded from below and above by $a_{m,i}$ and $a_{M,i},\ i \in \mathcal{V}$, respectively. Accordingly, for each $i \in \mathcal{V}$ the motion model reads as

$$\begin{aligned} \dot{v}_i(t) &= a_i(t), \\ \dot{s}_i(t) &= v_i(t) \end{aligned} \quad (2)$$

with the corresponding box constraints

$$\begin{aligned} a_{m,i} \leq \quad & a_i(t) && \leq a_{M,i}, \\ 0 \leq \quad & v_i(t) && \leq v_{M,i}, \end{aligned} \quad (3)$$

where $a_{m,i}$, $a_{M,i}$ are the control constraints, and $v_{M,i}$ denotes the legal speed limit, and the initial conditions

$$\begin{aligned} s_i(t_0) &= -s_{0,i}, \\ v_i(t_0) &= v_{0,i}, \end{aligned} \quad (4)$$

Table 1: Quantities and variables.

Symbol	Description	Sec/Fig
$i,j \in \mathcal{V}$	Vehicle indices with the corresponding index set	2.1
N	Number of vehicles	2.1
$k,l \in \mathcal{P}$	Path indices with the corresponding index set	2.2
M	Number of paths	2.2
$P(i)$	A path corresponding to vehicle i	2.2
c_{kl}	A k,l entry of a conflict matrix	2.2
$s_i(t)$	Position vehicle i along its path	3.1
$v_i(t)$	Velocity of vehicle i	3.1
$a_i(t)$	Acceleration (control variable) of vehicle i	3.1
t_i	Arrival time of vehicle i	3.1
$t_{i,min}, t_{i,max}$	Lower and upper bounds of t_i	3.1
$v_i(t_i)$	Arrival velocity of vehicle i	3.1
$T_i(t_i)$	An inverse of the arrival velocity of vehicle i	3.2
w_{ij}	Binary decision variable (equals to 1 if vehicle i passes first)	3.2
L^b_{ij}	Length to travel before overlapping zone of $P(i)$ and $P(j)$	3.2
L^a_{ij}	Length to travel to clear overlapping zone of $P(i)$ and $P(j)$	3.2
L_k	Length of path k in CZ	3.2

where $s_{0,i}$ are the distances left to the CZ staring at time t_0, and $v_{0,i}$ are the corresponding velocities at time t_0. Note that the terms $s_{0,i}$, $i \in \mathcal{V}$, are positive, and therefore the sign in front of them should be inverted.Vehicle geometrical characteristics can be summarized as follows. The footprint of each vehicle is represented by a rectangle. Each rectangle is conceived to contain actual planar dimensions of the corresponding vehicle. The locations given by the states $s_i(t)$, $i \in \mathcal{V}$, are actually the positions of the geometrical center of each rectangle. Since our global goal is to minimize the total time it takes for all vehicles to clear an intersection, and taking into account Assumption 6, we formulate the following OCPs as maximization of corresponding final velocities, i.e. the velocities at which vehicles finish the approaching phase, subject to vehicle model from above. For each $i \in \mathcal{V}$ we have

$$\max_{a_i(t)} \; v_i(t_i) \tag{5a}$$

$$\text{s.t.} \quad (2),(3),(4), \tag{5b}$$

$$s_i(t_i) = 0, \quad v_i(t_i) \le v_{P(i)}, \quad t \in [t_0, t_i], \tag{5c}$$

where t_i denotes the time at which the i-th car finishes its approaching phase (starts its intersection phase), and $v_{P(i)}$ is the upper bound on the final velocity. This upper bound depends on the maximum curvature of the corresponding path $P(i) \in \mathcal{P}$. Due to simplicity of the underlying model, OCPs (5) can be solved analytically by treating the times t_i, $i \in \mathcal{V}$, as parameters. By doing so, one obtains final velocities as functions of final times

$$v_i(t_i), \quad t_i \in [t_{i,min}, t_{i,max}], \quad i \in \mathcal{V}, \tag{6}$$

where $t_{i,min}$ denotes the minimal time when the i-th car can start the intersection phase, and $t_{i,max}$ denotes some reasonable time at which we are sure that the i's car will enter the intersection. In Section 3.2 we work with reciprocal of these functions.

On the one hand, there exists no technique how to determine sharp upper limits $t_{i,max}$, $i \in \mathcal{V}$, and they should be chosen as a good guess. However, note that starting from some $t_i^\star \in [t_{i,min}, t_{i,max}]$, $i \in \mathcal{V}$, the corresponding velocities will be constant. The lower limits $t_{i,min}$, $i \in \mathcal{V}$, on the other hand, can be determined as a minimal time for which problem (5) becomes feasible.

In this work we restrict ourselves to final velocity optimization for each vehicle, however this idea can be extended to any parametric function of t_i coming from an analytical solution of some underlying OCP. Moreover, if no analytical solution can be found, one can use a value function approach as e.g. in (Hult et al., 2018).

3.2 Scheduling Constraints

The goal of scheduling constraints is to ensure that the vehicle footprints at least do not overlap in a CZ. By saying "at least" we mean that we also have to preserve some *safety margin* between any pair of cars whose corresponding paths overlap. Overlaps of vehicle footprints together with a safety margin form a so-called *overlapping zone*. Possible choices of overlapping zones are discussed in the sequel.

As one can easily judge, scheduling constraints are safety critical. First, we define these constraints, and then we give a comprehensive explanation on how they work. The scheduling constraints read as

$$\begin{array}{rcl} t_i + L^a_{ij}T_i(t_i) - t_j - L^b_{ji}T_j(t_j) & \le & C_{big}(1-w_{ij}), \\ t_j + L^a_{ji}T_j(t_j) - t_i - L^b_{ij}T_i(t_i) & \le & C_{big}w_{ij}, \\ w_{ij} & \in & \{0,1\}, \\ \forall i,j \in \mathcal{V} \quad \text{with} \quad c_{P(i)P(j)} & = & 1, \end{array} \tag{7}$$

where

$$T_i(t_i) = 1/v_i(t_i), \quad t_i \in [t_{i,min}, t_{i,max}], \quad i \in \mathcal{V}.$$

The constraints given above are the so-called "either-or" constraints (Winston and Goldberg, 2004), due to the fact that the variables w_{ij}, $i,j \in \mathcal{V}$, can take only binary values and the constant C_{big} is assumed to be at least as large as $\sum_{i=1}^{N}[t_i + T_i(t_i)]$. Since the latter quantity is never known in advance, one can choose just some reasonably large constant.

So, let us now discuss what constraints (7) actually mean. We consider only the cases where $c_{P(i),P(j)} = 1$ for some $i,j \in \mathcal{V}$, i.e. there is a conflict between the paths $P(i)$ and $P(j)$, otherwise there is just no scheduling constraint. Without loss of generality let us assume that some $w_{ij} = 1$, which actually means that the i-th car goes before the j-th one, then the first line in (7) reads as

$$t_i + L^a_{ij}T_i(t_i) \le t_j + L^b_{ji}T_j(t_j), \tag{8}$$

where L^a_{ij} denotes the length of the path $P(i)$ which i-th vehicle has to travel in a CZ in order to leave the overlapping zone with the path of j-th vehicle. The term L^b_{ji}, in turn, denotes the length of the path $P(j)$ which the j-th vehicle has to travel in a CZ before entering the overlapping zone with the path of i-th vehicle. For a better understanding of the quantities discussed above see Fig. 2. Inequality (8) essentially states that the i-th vehicle has to leave the overlapping zone before the j-th vehicle enters it. An analogous argument holds for the second inequality in (7).

There are different ways how to define an overlapping zone. Let us discuss two limiting cases. The first one is when an overlapping zone is defined in a way such that vehicle footprints do not overlap and there is no additional safety margin, in other words, vehicles can touch each other at one single point. Of course, such a choice is not appropriate in any real-world situation just due to obvious safety issues. So, in a real-world scenario we need to preserve some safety margin. Another limiting case is when a safety margin is the whole CZ. In this case constraints (7) read as

$$\begin{array}{rcl} t_i + L_{P(i)}T_i(t_i) - t_j & \le & C_{big}(1-w_{ij}), \\ t_j + L_{P(j)}T_j(t_j) - t_i & \le & C_{big}w_{ij}, \\ w_{ij} & \in & \{0,1\}, \\ \forall i,j \in \mathcal{V} \quad \text{with} \quad c_{P(i)P(j)} & = & 1, \end{array} \tag{9}$$

where $L_{P(i)}$ and $L_{P(j)}$ denote the lengths of the $P(i)$'s and $P(j)$'s paths, respectively, see Fig. 2.

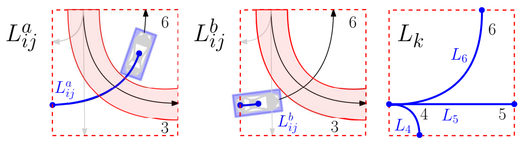

Figure 2: Left and middle: CZ representation with the quantities L^a_{ij} and L^b_{ij}, respectively. Right: paths $4,5$ and 6 with the corresponding lengths L_k.

3.3 MINLP

Now, we are at a position to formulate a mixed integer nonlinear minimization problem. As already mentioned before, our goal is to minimize the time at which the last vehicle in our consideration will clear an intersection. Note that we do not know in advance which vehicle will be the last one. Therefore, the objective function is given as a total sum of individual maneuver durations. The final mixed integer nonlinear problem is given by

$$\begin{array}{l} \min\limits_{t_i, w_{ij}} \sum\limits_{i=1}^{N} \left[t_i + L_{P(i)}T_i(t_i)\right], \\ \text{s.t.} \quad (7). \end{array} \tag{10}$$

Note that the problem stated above is a nonlinear problem, and most of existing software packages, can handle only linear problems. Therefore, we have to first linearize the problem, namely the nonlinearities $T_i(t_i)$, $i \in \mathcal{V}$, in a way such that a mixed integer linear problem solver can deal with it. One of possible linearization techniques is discussed in Section 4.

Once the optimal starting times t_i, $i \in \mathcal{V}$, are found, we can do a *post-regularization* step for the vehicles whose final velocities are saturated by $v_{i,M}$, $i \in \mathcal{V}$. The reason is obvious. If there is no saturation effect, vehicle's motion is fully described by (5). Therefore, there are no additional degrees of freedom. In the other case, however, there are infinitely many ways how to reach the CZ at the prescribed (by the MINLP solution) time. Here, we choose a simple (but the most comfortable) motion, namely we minimize the L^2-norm of the control variable. The corresponding OCP problem reads as

$$\begin{array}{l} \min\limits_{a_i(t)} \frac{1}{2}\int_{t_0}^{t_i^{opt}} a_i^2(t)dt, \\ \text{s.t.} \quad (5b),(5c), \\ v_i(t_i^{opt}) = v_{P(i)}, \end{array} \tag{11}$$

where in (5c) one has to substitute t_i by t_i^{opt}, which denotes the optimal staring time (in the global sense), i.e. t_i^{opt} comes from the solution of problem (10).

4 PIECEWISE LINEARIZATION

As mentioned at the end of Section 3.3, we need to find some linearization technique which would allow us to reformulate problem (10) into a linear problem. A straightforward idea would be to use piecewise linearizations of the nonlinearities $T_i(t_i)$, $i \in \mathcal{V}$, on some discretizations of the intervals $[t_{i,min}, t_{i,max}]$, $i \in \mathcal{V}$.

Unfortunately, existing high-performance software packages, like for instance GUROBI, which we use for our computations, do not directly accept piecewise linear functions, and therefore, we have to introduce some auxiliary variables and constraints. The method proposed in this paper is inspired by the linearization technique from (Winston and Goldberg, 2004).

Here, we also have to mention that there exist other, more sophisticated methods of solving MINLP: SQP-type algorithms, see e.g (Exler et al., 2012), or adaptive algorithms, see (Geißler et al., 2012), (Burlacu et al., 2020). The latter technique is very similar to the one we exploit in this work, however due to space restrictions, we do not consider those algorithms in the underlying work.

The idea behind the piecewise linearization according to (Winston and Goldberg, 2004) is to use convex combinations of nonlinear function values at discretization points, and to let the solver choose the right segment (where the time variable attains its optimal values in the global sense) by introducing some auxiliary binary variables. As we already did before, let us first give the formulas, and then discuss them. For each $i \in \mathcal{V}$ we have

$$t_i = \sum_{k=1}^{K_i} \lambda_{i,k}\, t_{i,k}, \tag{12a}$$

$$T_{i,pw} = \sum_{k=1}^{K_i} \lambda_{i,k}\, T_i(t_{i,k}), \tag{12b}$$

$$\sum_{k=1}^{K_i} \lambda_{i,k} = 1, \quad \sum_{k=1}^{K_i-1} z_{i,k} = 1, \tag{12c}$$

$$\begin{array}{rcl} \lambda_{i,1} & \leq & z_1, \\ \lambda_{i,k} & \leq & z_{i,k} + z_{i,k-1}, \quad k \in \{2,\dots,K_i-1\}, \\ \lambda_{i,K_i} & \leq & z_{K_i-1}, \end{array} \tag{12d}$$

$$\begin{array}{rcl} \lambda_{i,k} & \geq & 0, \quad k \in \{1,\dots,K_i\}, \\ z_{i,k} & \in & \{0,1\}, \quad k \in \{1,\dots,K_i-1\}, \end{array} \tag{12e}$$

where $t_{i,k}$, $k \in \{1,\dots,K_i\}$, are some grid points (not necessarily equidistant) in the interval $[t_{i,min}, t_{i,max}]$ with $t_{i,1} = t_{i,min}$ and $t_{i,K_i} = t_{i,max}$, and the constant K_i denotes the number of discretization points of the corresponding time interval. The idea behind (12) is the following. We introduce the real variables $\lambda_{i,k}$, $k \in \{1,\dots,K_i\}$, (lambdas), and the binary variables $z_{i,k}$, $k \in \{1,\dots,K_i-1\}$, (z-variables), where $i \in \mathcal{V}$. The role of the z-variables is to indicate the segment with the optimal values $t_i \in [t_{i,min}, t_{i,max}]$, $i \in \mathcal{V}$. Indeed, since those variables can take only binary values and their sum should be equal to one, only one z-variable will be non-zero. In turn, the role of the lambdas is nothing else but to build a convex combination over the segment, at which the non-zero z-variable is pointing. This relation is given by (12d), which also guarantees that only two lambdas are non-zero. Additionally, the first sum in (12c) guarantees that the sum of these two lambdas is one. Therefore, equalities (12a) and (12b) are linear combinations over the "optimal" segment.

4.1 MILP Problem

Finally, we can formulate a mixed integer linear problem as an approximation of (10). It reads as follows

$$\begin{array}{l} \min\limits_{t_i, w_{ij}} \sum\limits_{i=1}^{N} \left[t_i + L_{P(i)} T_{pw,i} \right], \\ \text{s.t.} \quad (7), (12). \end{array} \tag{13}$$

We solve problem (13) with the GUROBI MILP-solver (Gurobi Optimization, 2020) which is using a linear-programming based branch-and-bound algorithm. The next section is devoted to examples and conclusions.

5 SIMULATIONS

In order to validate the proposed algorithm, we performed simulations in different scenarios, namely, we consider the following intersection geometries: 8-lanes intersection, 16-lanes intersection and 10-lanes irregular intersection as depicted in Fig. 1, 3 and 4, respectively. For each intersection geometry we construct the corresponding scenario such that it represents some complex traffic situation in which right-of-way rule or first-arrive-first-served approaches are outperformed by our approach. For a better comprehension of all scenarios considered in this paper we encourage you to watch a short video (Gerdts et al., 2020). Moreover, in order to show the robustness of the algorithm, for the latter intersection geometry we perform 24 simulations with different initial conditions. In all the upcoming simulations identical vehicles are considered with $a_{m,i} = -5\ m/s^2$, $a_{M,i} = 3\ m/s^2$ and $v_{M,i} = 17\ m/s$, $i \in \mathcal{V}$. All simulations are performed on an Intel quad-core i7 2,9 GHz processor. Before we dive into simulations, we would like to discuss how grid points for the piecewise linearization of the corresponding nonlinearities $T_i(t_i)$, $i \in \mathcal{V}$, are chosen. Obviously, we want to minimize the approximation error. In order to do so, some grid points have to correspond to stationary and kink points of the nonlinear functions.This ensures both that all values of the nonlinear functions are represented by corresponding piecewise linear functions and that the approximation is exact for the portions of the nonlinear functions which correspond to saturations. Note thah those portions require only two grid points in order to be exact. In the simulations we use 10 grid points, leading to an approximation error less than one millisecond.

8-lanes Intersection: In this example we consider the whole conflict zone as an overlapping zone for each pair of conflicting paths, i.e. scheduling constraints are given by (9). The initial conditions and optimal times can be found in Table 2, and the corresponding trajectories in Fig. 5. Vehicles 3 and 4 go before vehicles 1 and 2 because the latter ones are subject to velocity limitations due to corresponding maximal curvatures even though the first pair starts closer than the second one. Also, the right-turning vehicle goes before the left-turning one, since the right-turning vehicle's path is shorter than the one of the other vehicle, even though the left-turning vehicle can perform its maneuver at a higher velocity.

16-lanes Intersection: This example is constructed in a way such that it is hard to guess an optimal order in advance. Moreover, with this example we show scalability of the proposed algorithm, since the solver runtime is very similar to the previous case. The intersection geometry is depicted in Fig. 3. The initial conditions and optimal starting times are given in Table 2. In this case we define proper overlapping zones by considering the same footprint for each vehicle with the length of 4 m and the width of 2 m. The trajectories can be found in Fig. 5. The optimal order is similar to the previous one. Four vehicles, that have straight paths, go first. The other four vehicles have to perform pairwise symmetric maneuvers, since there is not enough room in the intersection to cross it simultaneously.

10-lanes Irregular Intersection: With this example we want to demonstrate the performance of the algorithm in a very safety-critical situation at an irregular intersection. Corresponding geometry, initial conditions and optimal times as well as optimal trajectories can be found in Fig. 4, Table 2 and Fig. 5, respectively. Also, for such kind of intersection we performed 24 simulations with 4 vehicles. We considered different combinations of paths and initial conditions. The average solver runtime was 0.0060 s with the maximum of 0.0187 s.

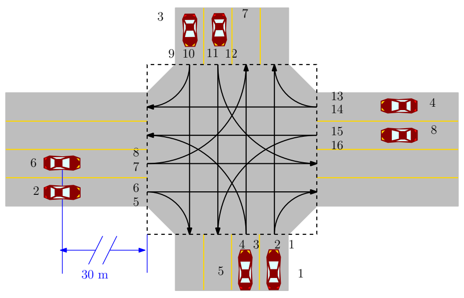

Figure 3: Geometry of the 16-lanes intersection.

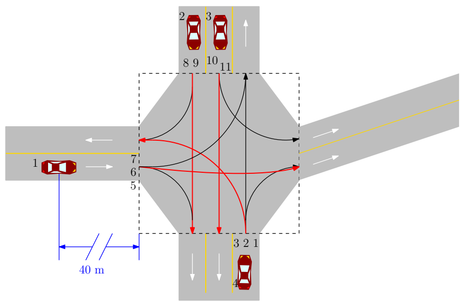

Figure 4: Geometry of the irregular 10-lanes.

6 CONCLUSIONS AND OUTLOOK

In this paper we proposed a bi-level optimization algorithm for an intersection management problem. On the lower level, for each vehicle we considered a simple OCP, which provided us optimal final velocities as nonlinear functions of final times. On the higher level, in turn, we exploited those functions in a MINLP. We solved that problem by approximating the nonlinearities with piecewise linear functions, which resulted in a more complex but linear problem. We performed numerical experiments using GUROBI MILP-solver to validate the algorithm, which showed that it is robust and fast. The emerging MINLP can be solved with a slightly different linearization technique inspired by (Burlacu et al., 2020), which gives a con-

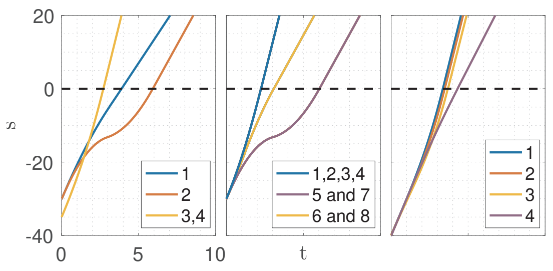

Figure 5: From left to right: 4-lanes, 16-lanes and 10-lanes solutions trajectories.

Table 2: Initial conditions and optimal times for the 8-lanes, 16-lanes and 10-lanes intersections, respectively.

i	$P(i)$	$s_{0,i}$	$v_{0,i}$	t_i	$L_{P(i)}/v_i$
1	7	30 m	10 m/s	3.883 s	2.031 s
2	3	30 m	10 m/s	5.914 s	2.436 s
3	5	35 m	10 m/s	2.706 s	1.176 s
4	11	35 m	10 m/s	2.706 s	1.176 s
Solver runtime			0.0040 s		
1	2	30 m	10 m/s	2.244 s	1.195 s
2	6	30 m	10 m/s	2.244 s	1.195 s
3	10	30 m	10 m/s	2.244 s	1.195 s
4	14	30 m	10 m/s	2.244 s	1.195 s
5	4	30 m	10 m/s	6.033 s	2.436 s
6	8	30 m	10 m/s	3.071 s	2.436 s
7	12	30 m	10 m/s	6.033 s	2.436 s
8	16	30 m	10 m/s	3.071 s	2.436 s
Solver runtime			0.012 s		
1	6	40 m	10 m/s	3.354 s	1.176 s
2	9	40 m	10 m/s	3.478 s	1.192 s
3	10	40 m	10 m/s	3.680 s	1.213 s
4	3	40 m	10 m/s	4.307 s	2.834 s
Solver runtime			0.0072 s		

trol over the error between the nonlinearities and their approximations. Such linearization technique can be more efficient in the case of nonlinear functions with high derivatives.

Our algorithm can be applied to any car model, even with uncertainties. Just in case if no analytical solution of the corresponding OCP can be found, one can exploit a value function approach like in (Hult et al., 2018). Finally, we would like to announce that in one of our future works we are going to consider a more general case with multiple cars in one lane, where a deep analysis of rear-end collision avoidance conditions has to be performed.

REFERENCES

Ahmane, M., Abbas-Turki, A., Perronnet, F., Wu, J., El Moudni, A., Buisson, J., and Zeo, R. (2013). Modeling and controlling an isolated urban intersection based on cooperative vehicles. *Transportation Research Part C: Emerging Technologies*, 28:44–62.

Britzelmeier, A. and Dreves, A. (2020). A decomposition algorithm for Nash equilibria in intersection management. *Optimization*, pages 1–38.

Burlacu, R., Geißler, B., and Schewe, L. (2020). Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. *Optimization Methods and Software*, 35(1):37–64.

Exler, O., Lehmann, T., and Schittkowski, K. (2012). A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization. *Mathematical Programming Computation*, 4(4):383–412.

Fayazi, S. A. and Vahidi, A. (2018). Mixed-Integer Linear Programming for Optimal Scheduling of Autonomous Vehicle Intersection Crossing. *IEEE Transactions on Intelligent Vehicles*, 3(3):287–299.

Geißler, B., Martin, A., Morsi, A., and Schewe, L. (2012). Using piecewise linear functions for solving minlps. In *Mixed integer nonlinear programming*. Springer, New York.

Gerdts, M., Rogovs, S., and Valenti, G. (2020). Optimal scheduling of autonomous cars at an intersection. https://youtu.be/uzZGm8WOFRA.

Gurobi Optimization, L. (2020). Gurobi optimizer reference manual.

Hult, R., Zanon, M., Frison, G., Gros, S., and Falcone, P. (2020). Experimental validation of a semi-distributed sequential quadratic programming method for optimal coordination of automated vehicles at intersections. *Optimal Control Applications and Methods*, 41(4):1068–1096.

Hult, R., Zanon, M., Gras, S., and Falcone, P. (2018). An miqp-based heuristic for optimal coordination of vehicles at intersections. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE.

Hult, R., Zanon, M., Gros, S., and Falcone, P. (2019). Optimal Coordination of Automated Vehicles at Intersections with Turns. In *2019 18th European Control Conference (ECC)*. IEEE.

Khayatian, M., Mehrabian, M., Andert, E., Dedinsky, R., Choudhary, S., Lou, Y., and Shirvastava, A. (2020). A Survey on Intersection Management of Connected Autonomous Vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(4):1–27.

Malikopoulos, A. A., Cassandras, C. G., and Zhang, Y. J. (2018). A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections. *Automatica*, 93:244–256.

Mohamad Nor, M. H. and Namerikawa, T. (2019). Optimal Coordination and Control of Connected and Automated Vehicles at Intersections via Mixed Integer Linear Programming. *SICE Journal of Control, Measurement, and System Integration*, 12(6):215–222.

Namazi, E., Li, J., and Lu, C. (2019). Intelligent Intersection Management Systems Considering Autonomous Vehicles: A Systematic Literature Review. *IEEE Access*, 7:91946–91965.

Winston, W. L. and Goldberg, J. B. (2004). *Operations research: applications and algorithms*. Thomson/Brooks/Cole, Belmont, CA.