
CHALMERS
UNIVERSITY OF TECHNOLOGY

# Intersection crossing with reduced number of conflicts

2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

# Intersection crossing with reduced number of conflicts

Johan Karlsson, Jonas Sjöberg, Nikolce Murgovski,
Lowisa Hanning, Susan Luu, Vanessa Olsson and Alexander Rasch

***Abstract*— In this paper, an optimization based algorithm for safe and efficient collaborative driving in intersections is formulated. The problem is to determine the optimal order in which vehicles should travel through an intersection under the assumptions that the longitudinal velocity of all vehicles can be controlled along a predefined path. In the original formulation one quadratic optimization program was solved for each possible crossing order of the vehicles and collisions were avoided by formulating constraints that only allowed one vehicle at the time inside the intersection. To make this algorithm more effective, we formulate less restrictive collision avoidance constraints by introducing one critical zone for each point where two predefined paths cross. It is shown that this formulation leads to a decrease in the number of quadratic optimization programs that need to be solved to find the best crossing order. Further, an algorithm is provided that finds the number of crossing sequences which yield unique formulations of the optimization program. The results show that when simulating more complex scenarios, like four vehicles traveling through an ordinary intersection, the reduction of computational time and the total time it takes for all vehicles to make it through the intersection can be significantly reduced using these less restrictive constraints.**

## I. Introduction

The development of Intelligent transportation systems (ITS) has been a major subject for both the automotive industry as well as governmental institutions in recent year [1]. This due to its potential for safer, smarter and greener solutions [2]. Autonomous driving and active safety developments are two directions undergoing extensive research.

In this paper we will study cooperative autonomous driving in intersections. Intersections are interesting from the perspective of autonomous driving and active safety since the risk for accidents are high. In Europe, intersection-related accidents are responsible for $21\%$ of traffic related deaths and $43\%$ of non-fatal injuries [3]. Similar numbers have been reported for the U.S. [4]. Due to this high accident risk, intersections are highly regulated by traffic lights, signs and road markings. This might decrease the number of accidents, but instead turn intersections into bottlenecks which increase congestion.

The problem of autonomously controlling a fleet of vehicles through an intersection is two-fold. In which order should the vehicles drive through the crossing and what speeds and accelerations should the vehicles have when doing so. Several cooperative conflict resolution techniques have been suggested, including scheduling formulations [5], [6], multi-agent approaches [7], [8] and model predictive control (MPC) both centralized [9], [10] and desentralized [11], [12], [13]. In this paper we will build upon the collaborative driving approach presented in [9], [10], in which a centralized MPC coordination strategy is formulated to solve the intersection problem. In this formulation the optimal control programs is a quadratic program (QP) for all permutations of crossing sequences. In each of these QPs collision avoidance is assured by introducing constraints that allow only one vehicle to reside within a critical zone at a time. In [9] the critical zone is simply chosen to be the full intersection. In this paper we will, based on the same idea as in [11], focus on modifying the optimal control problem to use less conservative constraints for collision avoidance. However, as we shall see, doing so using the approach in [9] compared to in [11] will provide a convex optimization program instead of a non-convex one.

Further, in [9] the crossing order of vehicles was decided by solving a QP for every possible crossing order and choosing the one that gave the smallest cost. In this paper, we will show that when introducing less restrictive collision avoidance, it is, for many scenarios, enough to solve a subset of the QPs to find (one of) the optimal crossing orders. An algorithm is presented which calculates such a subset of QPs.

The paper is organized as follows. In Section II the QP formulation, first presented in [9], is given. Then, in Section III, we present the QP suggested in this paper, which includes a reformulation of the collision constraints. Section IV continues to explore this new formulation by showing that it is enough to solve a subset of the optimization programs, to guarantee the optimal crossing sequence. It also introduces an algorithm to find this subset of optimization programs. Section V proves that the algorithm introduced in the previous section in fact results in the minimum possible number of crossing orders that guarantee a solution. Then, a case study is presented in Section VI which compare the novel approach with the original one, before conclusions are drawn in Section VII.

## II. Problem formulation

Consider $N > 1$ number of autonomous vehicles approaching an intersection. The aim is to decide the crossing order and longitudinal acceleration of each vehicle. This will be done via a centralized MPC algorithm which is assumed to have the data of all vehicles available. The algorithm operates under the assumption that vehicle $i$

1) has a position trajectory is given and known;
2) has a given path is perfectly followed;

This work was partially supported by the Wallenberg Autonomous Systems Program (WASP) and partially by the European Commission Seventh Framework Program under the project AdaptIVe, grant agreement number 610428.

3) has a clock synchronized with the other vehicles;
4) is located within the radius of a circle centered at the intersection.

The case where several vehicles approach or leave the intersection on the same intersection leg, i.e, following each other outside of the are of intersection, is for readability, not considered in this paper.

### A. Vehicle dynamics

Each vehicle $i$, following a known path $p$, is modeled as a point mass using the state vector $\mathbf{x}_i(p) = [t_i(p), z_i(p)]^T$, where $t_i(p)$ is the traveling time of vehicle $i$ and $z_i(p)$ is the inverse of vehicle $i$'s velocity along the path. from here on $z_i(p)$ is refered to as the lethargy of vehicle $i$. If we choose the input signal to be the spatial derivative of the lethargy, $u_i(p) = z_i'(p)$, the vehicles dynamics are described by the linear system

$$\mathbf{x}_i'(p) = A\mathbf{x}_i(p) + Bu_i(p), \quad \forall i \in \mathcal{N} = \{1, \dots, N\},$$

where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

### B. State and actuator limits

The state and control variables are subject to lower and upper bounds. To account for curves, the maximum velocity of the ego vehicle is set to $v_{i\text{max}} = \min(v_\text{l}, \sqrt{a_{i\text{ymax}} r_i(p)})$ where $v_\text{l}$ is the speed limit of the road, $r_i(p)$ is the radius of vehicle $i$'s planned path and $a_{iy\text{max}}(p)$ is the maximum lateral velocity along the path. The minimum and maximum bounds of the states are then written as

$$\mathbf{x}_{i\text{min}}(p) = \begin{bmatrix} \int_0^p \frac{1}{v_{i\text{max}}(s)} ds \\ \frac{1}{v_{i\text{max}}(p)}, \end{bmatrix}, \ \mathbf{x}_{i\text{max}}(p) = \begin{bmatrix} \int_0^p \frac{1}{v_{i\text{min}}(s)} ds \\ \frac{1}{v_{i\text{min}}(p)} \end{bmatrix}$$

where $v_{i\text{min}}(p), v_{i\text{max}}(p) > 0$ is the minimum and maximum longitudinal velocity of vehicle $i$, respectively. For the limits on the control signal we choose the linearized bounds derived in [9], which read

$$u_\text{min}(\cdot) = a_{i\text{max}}(p)(2 - 3v_{i\text{r}}(p)z_i(p))/v_{i\text{r}}^3(p),$$
$$u_\text{max}(\cdot) = a_{i\text{min}}(p)(2 - 3v_{i\text{r}}(p)z_i(p))/v_{i\text{r}}^3(p),$$

where $v_{ir}$ is the reference velocity of vehicle $i$ and the velocity around which the bounds are linearized.

### C. Collision avoidance

In [9] it is proposed to impose non-collision constraints by treating the intersection as a critical set, meaning that only one vehicle is allowed to reside within the intersection at any given time. The set of positions for vehicle $i \in \mathcal{N}$ inside the critical zone, is called the critical set and can be formulated as

$$\mathcal{C}_i = \{p_i(t) \in [0, p_{i\text{f}}] | p_i(t) \in [L_i, H_i]\},$$

where $p_{i\text{f}}$ is the final position of vehicle $i$. Further, the corresponding times can be expressed in terms of the set

$$\mathcal{G}_i(u_i(t)) = \{t \in [0, t_{i\text{f}}] | p_i(t) \in \mathcal{C}_i\},$$

where $t_{if}$ is the time when vehicle $i$ reaches its final position $p_{i\text{f}}$.

### D. Cost function

In [9], several different cost functions are suggested, depending on the prefered behavior of the solution. In this paper, we stick to the simple choice of minimizing the distance from a known velocity trajectory and minimizing the actuator limits for each vehicle. further, this cost function can be approximated by a quadratic cost fuction by replacing the reference velocity $v_{i\text{r}}(p)$ with the mean reference velocity $\bar{v}_{i\text{r}}(p)$ and linearize the first integrand around it. Doing so yields

$$J_i(\cdot) \approx w_{i1} \int_0^{p_{i\text{f}}} \left( z_i(p) - \frac{1}{v_{i\text{r}}(p)} \right)^2 dp + w_{i2} \bar{v}_{i\text{r}}^5 \int_0^{p_{i\text{f}}} u_i^2(p) dp + w_{i3} \bar{v}_{i\text{r}}^7 \int_0^{p_{i\text{f}}} u_i'^2(p) dp = \tilde{J}_i(\cdot).$$

The cost function used in the optimization program is the sum of the individual cost functions of the $N$ vehicles, i.e.,

$$J(\cdot) = \sum_{i \in \mathcal{N}} \tilde{J}_i(\mathbf{x}_i(p), u_i(p), u_i'(p), \mathbf{x}_i(p_{i\text{f}})).$$

See [9] for more details.

### E. Optimal control problem

Let the matrix of all possible crossing orders be $\mathcal{O} \in \mathcal{N}^{M \times N}$, where $M = N!$ is the number of crossing orders and $N$ is the number of cars. Then, the task of determining the optimal longitudinal behavior of $N$ vehicles moving through the crossing in a given crossing order $m$ can be written as

$$\underset{u_\text{E}(t)}{\text{minimize}} \ J(\mathbf{x}_i(p), u_i(p), u_i'(p), \mathbf{x}_i(p_{i\text{f}})) \tag{1a}$$

subject to

$$\mathbf{x}_i'(p) = A\mathbf{x}_i(p) + Bu_i(p), \quad \forall i \in \mathcal{N}, \tag{1b}$$
$$\mathbf{x}_i(p) \in [\mathbf{x}_{i\text{min}}(p), \mathbf{x}_{i\text{max}}(p)], \quad \forall i \in \mathcal{N}, \tag{1c}$$
$$u_i(p) \in [u_\text{min}(p, z_i(p)), u_\text{max}(p, z_i(p))], \ \forall i \in \mathcal{N}, \tag{1d}$$
$$\mathbf{x}_i(0) = \mathbf{x}_{i0}, \mathbf{x}_i(p_{i\text{f}}) = \mathbf{x}_{i\text{f}}, \quad \forall i \in \mathcal{N}, \tag{1e}$$
$$t_k(H_k) \leq t_l(L_l), k = \mathcal{O}_{m,n}, l = \mathcal{O}_{m,n+1}, \forall n \in \mathcal{N} \setminus N. \tag{1f}$$

Thus, the best crossing order is found by solving (1) for all $m = 1, \dots, M$ and choosing the one with the lowest cost. In Section III the collision constraints 1f will be replaced by less restrictive once.

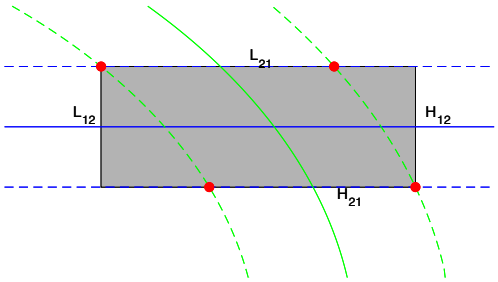


Fig. 1. Illustration of a shared zone between two vehicles. The shared zone is marked in grey, $L_{ij}$ and $H_{ij}$ are entry and exit for vehicles $j = 1, 2$, of the shared zone. The solid blue and green lines show the predefined paths and the dashed show the limits of the vehicles.

## III. Defining less conservative critical zones

Recall that in the initial problem formulation, (1), one critical zone was introduced, which, applied for all vehicles. In this section we suggest a different way of defining the non-collision constraints which are less restrictive and allow for faster traffic flow through the intersection. This is done by replacing the critical set with a set of *shared zones*. A shared zone is a limited rectangular area within the intersection where two (or more) vehicles can reside at the same time, according to their predefined paths, see Fig. 1. The minimum size of the shared zones are decided by the intersection of the left and right safety margins of the vehicle (illustrated by red dots in 1), which in turn define the entry and exit points $L_{ij}$ and $H_{ij}$ for vehicles $i$ and $j$. The set of positions for which the vehicle $i \in \mathcal{N}$ resides within a shared zone with the vehicle $j$ can be expressed as

$$\mathcal{C}_{ij} = \{p_i(t) \in [0, p_{if}] | p_i(t) \in [L_{ij}, H_{ij}]\}.$$

From this set we define the set of vehicle pairs that do not have a shared zone as $\mathcal{I} = \{(i, j) \in \mathcal{N} \times \mathcal{N} | \mathcal{C}_{ij} = \emptyset\}$.

### A. Full program using shared zones

This leads to a new optimization program

$$\underset{u_E(t)}{\text{minimize}}\ J(\mathbf{x}_i(p), u_i(p), u_i'(p), \mathbf{x}_i(p_{if})) \tag{2a}$$

subject to

(1b) – (1e)

$$t_k(H_{kl}) \leq t_l(L_{lk}), k = \mathcal{O}'_{m,n}, l = \mathcal{O}'_{m,n_2}, \quad n_1 < n_2 \text{ and } (k, l) \in \mathcal{I}^c, \forall n_1 \in \mathcal{N}. \tag{2b}$$

where $\mathcal{I}^c$ denotes the set complement of $\mathcal{I}$. The non-collision constraints (2b) check the times when the vehicles enter and leave a shared zone instead of the intersection. In contrast to the non-collision constraints (1f) it is not enough to check for constraints between consecutive vehicles in the crossing order but constraints need to be imposed between all vehicles crossing paths. Further, the crossing order matrix $\mathcal{O}$ has been replaced by a new matrix $\mathcal{O}'$ of crossing orders. This is because, when introducing the shared zones, it is clear that the number of interesting crossing orders might be reduced. For instance, for the scenario depicted in Fig. 2, which will be discussed in Section IV, the optimization program (2) will yield the same solution regardless of where in the crossing order the blue vehicle is located, since it does not have any constraints related to the other vehicles. In Section IV we will present an algorithm that construct the matrix $\mathcal{O}'$.

## IV. Reduction of number of optimization problems solved

In this section an algorithm is defined that finds the matrix of reduced crossing orders $\mathcal{O}'$. The idea behind the algorithm is to remove crossing orders that lead to identical collision constraints (2b). Two collision orders $\mathcal{O}_{m,:}$ and $\mathcal{O}_{k,:}$ lead to indentical collision constraints if the only difference between them is that consecutive vehicles, which have no shared zone (i.e., do not impose a constraint), have switched places.

**Example 1.** *Consider the scenario depicted in Fig. 2. For this case, the matrix of all crossing orders is*

$$\mathcal{O} = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]^T. \tag{3}$$

*It is clear that vehicle* 1 *does not have a shared zone with vehicle* 2 *or* 3*. Thus, the consecutive order of vehicles* $1, 2$ *and* $1, 3$ *does not matter. This can be seen by studying the constraints the crossing orders give rise to*

$$\begin{aligned}
&(1,2,3) \Rightarrow t_2(H_{23}) \leq t_3(L_{32}), (1,3,2) \Rightarrow t_3(H_{32}) \leq t_2(L_{23}) \\
&(2,1,3) \Rightarrow t_2(H_{23}) \leq t_3(L_{32}), (2,3,1) \Rightarrow t_2(H_{23}) \leq t_3(L_{32}) \\
&(3,1,2) \Rightarrow t_3(H_{32}) \leq t_2(L_{23}), (3,2,1) \Rightarrow t_3(H_{32}) \leq t_2(L_{23})
\end{aligned}$$

*As can be seen, crossing orders* (1,2,3),(2,1,3),(2,3,1) *give rise to identical constraints. Further, this is so since the only difference between* (1,2,3) *and* (2,1,3) *is that the vehicles* 1 *and* 2*, which do not cross paths, have swapped places. The same is true for* (2,1,3) *and* (2,3,1)*, where non-crossing vehicles have switched places. Lastly, note that* (1,3,2),(3,1,2),(3,2,1) *yield identical constraints for similar reasons. Thus, in this specific scenario the number of optimization programs needed to be solved can be reduced from* 6 *to* 2*.*

Based on the idea that crossing orders are identical under the swap of non-crossing vehicles we formulate Algorithm (IV.1) to calculates the matrix of reduced crossing orders $\mathcal{O}'$.

**Algorithm IV.1.**

Step 0: *Find the set* $\mathcal{I}_1 = \{(i, j) \in \mathcal{I} : i < j\}$*, sort it on the first and then second entry. Further, create the empty set* $\mathcal{I}_2$*.*

Step 1: *Take the first element* $(i, j)$ *in* $\mathcal{I}_1$*.*

Step 2: *Go through the rows of the crossing order matrix* $\mathcal{O}$ *and remove all orders (rows) containing the pair* $(i, j)$ *in sequence. Also, set* $\mathcal{I}_2$=$\mathcal{I}_2 \cup \{j\}$*.*

Step 3: *Remove* $(i, j)$ *from* $\mathcal{I}_1$*, i.e,* $\mathcal{I}_1 = \mathcal{I}_1 \setminus (i, j)$*.*

Step 4: *If* $\mathcal{I}_1 = \emptyset$ *stop, otherwise take the next element* $(k, l)$ *in* $\mathcal{I}$ *and do the first of the following that apply*

*1) If* $k \in \mathcal{I}_2$ *set* $(i, j) = (l, k)$ *and go to step 2,*

*2) Set* $(i, j) = (k, l)$ *go to step 2.*

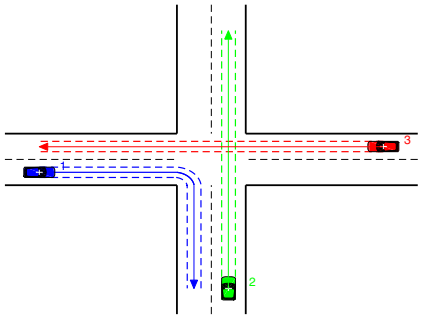


Fig. 2. Scenario with three vehicles, where the red and green cross paths.

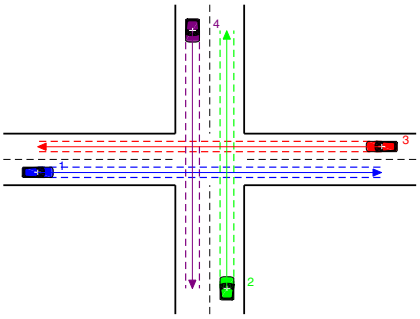


Fig. 3. Scenario 2, four vehicles all driving straight.

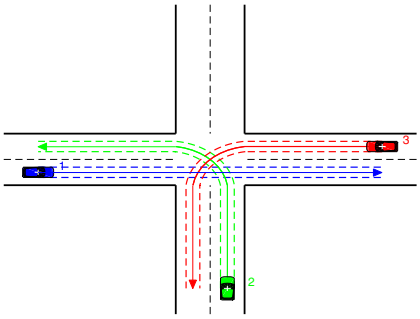


Fig. 4. Scenario 3, three vehicles, all crossing paths.

As is implied in Step 4 of Algorithm IV.1 it matters if crossing orders containing $(i,j)$ or $(j,i)$ is removed from $\mathcal{O}$. This is illustrated by revisiting the scenario in Fig. 2.

**Example 2.** *In Example 1 we saw that crossing orders* (1,2,3),(2,1,3),(2,3,1) *yield identical collision constraints. As did the three crossing orders* (1,3,2),(3,1,2),(3,2,1). *Hence, it is enough to solve two optimization program of the type* (2)*, using one crossing order from* (1,2,3),(2,1,3),(2,3,1) *and one from* (1,3,2),(3,1,2),(3,2,1). *Further, this means that Algorithm IV.1 should provide us with two elements, one from each group of crossing orders.*

*Apply Algorithm IV.1 to the original matrix of crossing orders,* (3)*. Set* $\mathcal{I} = \{$(1,3),(3,1),(1,2),(2,1)$\}$ *so the sets defined in Step 0 of the algorithm is* $\mathcal{I}_1 = \{$(1,2),(1,3)$\}$ *and* $\mathcal{I}_2 = \emptyset$*. Now, choose the first entry in* $\mathcal{I}_1$*, which is* (1,2)*, and remove all rows of* $\mathcal{O}$ *containing exactly that sequence, this results in the matrix* $[[1,3,2],[2,1,3],[2,3,1],[3,2,1]]^T$*. Further, we also set* $\mathcal{I}_2 = \{2\}$*. In the third step, remove* (1,2) *from* $\mathcal{I}_1$*, i.e., now* $\mathcal{I}_1 = \{$(1,3)$\}$*. In step four,* $\mathcal{I}_1$ *is not empty so take the element that is left,* (1,3)*, and note that* $1 \notin \mathcal{I}_2$*, so choose* (1,3) *and go back to step 2 which after completion gives* $[[2,3,1],[3,2,1]]^T$*. This is the end of the algorithm since* $\mathcal{I}_1$ *is now empty. As can be seen, the resulting matrix of crossing orders* $\mathcal{O}' = [[2,3,1],[3,2,1]]^T$ *contains exactly one element from* (1,2,3),(2,1,3),(2,3,1) *and one element from* (1,3,2),(3,1,2),(3,2,1)*. It was important that the second sequence removed was* (1,3) *and not* (3,1) *since removing* (3,1) *would have resulted in the following three rows in the reduced crossing matrix* (1,3,2),(2,1,3),(3,2,1).

As is evident from Examples 1 and 2 the minimum number of crossing orders are not unique. However, note that all the minimum number of crossing orders give the same solution. In the next section, Section V, it is proved that Algorithm IV.1 gives the minimum number of crossing orders under the assumptions that the intersection has $k$ legs, there are $n$ vehicles and at most one vehicle in each lane at all times.

## V. Proof of Algorithm

The aim of this section is to prove that Algorithm IV.1 provides us with the smallest possible reduced crossing order matrix which provides all unique formulations of the optimization program (2). The proof is divided into two main parts and follows the same outline as Examples 1 and 2. What this means is that the first part of the proof groups the crossing orders that yield identical collision constraints, similar to what is done in Example 1. This is done via the concept of equivalence relations and equivalence classes. For an introduction to these fundamental mathematical concepts, see, for instance [14], [15]. The second part of the proof is to apply the algorithm IV.1 to a general crossing scenario as we did in Example 2 and prove that we end up with exactly one crossing order from each group.

### A. Group crossing orders

For convenience of the proof, we will, instead of discussing the crossing order matrix $\mathcal{O}$ study the set of all crossing orders $\mathcal{S}$, i.e, the rows of $\mathcal{O}$ are the elements of $\mathcal{S}$. As mentiond, the idea of this part of the proof is to group the crossing orders using equivalence classes. The reason for this, is that there is a wellknown theorem in abstract algebra guaranteeing that equivalence classes on a set, in our case $\mathcal{S}$, forms a partition of that set. To define equivalence classes we first need to define an equivalence relation on $\mathcal{S}$. Further, to define this relation we define a bijection $\text{swap}_{(i,j)} : \mathcal{S} \to \mathcal{S}$ for each, non-crossing vehicle pair $(i,j) \in \mathcal{I}_1 = \{(i,j) \in \mathcal{I} : i < j\}$, such that if $s \in \mathcal{S}$

$$\text{swap}_{(i,j)}(s) = \begin{cases} s_{(j,i)} \text{ if } s = s_{(i,j)} \\ s_{(i,j)} \text{ if } s = s_{(j,i)} \\ s = s \text{ otherwise.} \end{cases} \tag{4}$$

where $s_{(i,j)}$ is an element of $\mathcal{S}$ which contains the sequence $(i,j)$. The swap operator (4) swap the places of vehicles $i,j$ in the crossing order if they appear in sequence, otherwise it leaves the crossing order unchanged. Further, we say that two elements $s_1, s_2 \in \mathcal{S}$ fulfill relation $R$ if

$$\begin{aligned} &s_1 R s_2 = \{(s_1,s_2) \in \mathcal{S} \times \mathcal{S} : \\ &\text{swap}^k_{(i,j)}(s_1) = s_2 \text{ for some } (i,j) \in \mathcal{I}_1 \text{ and } k = \{1,2\}\}. \end{aligned} \tag{5}$$

where $\text{swap}^2_{(i,j)}(s) = \text{swap}_{(i,j)}(\text{swap}_{(i,j)}(s))$. The interpretation of two vehicles being in the relation $R$ is that they lead to identical optimization programs, since either the permutations are the same, or we have shifted the position of vehicles that do not have a shared zone with each other. With the observation that any operator $\text{swap}_{(i,j)}$ with $(i,j) \in \mathcal{I}_1$

is its own inverse, it is easy to show that $R$ is an equivalence relation on $\mathcal{S}$.

**Theorem 1.** *The relation $R$ defined by* (5) *is an equivalence relation on the set of crossing orders $\mathcal{S}$.*

*Proof.* Let $s_1, s_2, s_3 \in \mathcal{S}$ and let $\text{swap}_{(i,j)}$ for some $(i,j) \in \mathcal{I}_1$.

Reflexive? Prove that $s_1 R s_1$. $s_1 \in \mathcal{S}$. True, since $\text{swap}^2_{(i,j)}(s_1) = s_1$ for any element in $\mathcal{S}$.

Symmetric? Prove that if $s_1 R s_2$ then $s_2 R s_1$. True, since if $\text{swap}_{(i,j)}(s_1) = s_2$ then $\text{swap}_{(i,j)}(s_2) = \text{swap}_{(i,j)}(\text{swap}_{(i,j)}(s_1)) = s_1$

Transitive? Prove that if $s_1 R s_2$ and $s_2 R s_3$ then $s_1 R s_3$. This is true since $\text{swap}^2_{(i,j)}(s_1) = \text{swap}_{(i,j)}(\text{swap}_{(i,j)}(s_1)) = \text{swap}_{(i,j)}(s_2) = s_3$. □

Using this equivalence relation we define the equivalence class on $\mathcal{S}$ of the element $s_1$ to be $[s_1] = \{s \in \mathcal{S} : s R s_1\}$. From the equivalence relation we obtain a family of equivalent classes $[s_1], [s_2] \dots, [s_k]$. According to a well-known theorem in abstract algebra equivalent classes of a set forms a partition of that set, [15]. Thus, the equivalence classes $[s_1], [s_2] \dots, [s_k]$ form a partition of $\mathcal{S}$, i.e. $\cup [s_i] = \mathcal{S}$ and $[s_i] \cap [s_j] = \emptyset$ if $i \neq j$. We make the claim that the elements within each equivalence class will lead to identical formulation of the optimization program (2), which is evident from the interpretation made earlier of the relation $R$. Thus, to obtain all unique formulations of the optimization program (2) it is enough to pick exactly one representative from each equivalence class. The goal is now, to prove that Algorithm IV.1 does exactly that. But first, let us return to scenario 2 to illustrate the mathematical concepts we introduced above.

**Example 3.** *Consider, again, the crossing scenario depicted in Fig. 2. For this case, the set of all crossing orders is*

$$\mathcal{S} = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}.$$

*Next, we want to define one operator for each vehicle pair which do not have crossing paths. As can be seen in Fig. 2, the vehicle* 1 *do not cross path with vehicle* 2 *or* 3*. Therefore, we introduce the swap operators* $\text{swap}_{(1,2)}$ *and* $\text{swap}_{(1,3)}$ *as defined by* (4)*. Applying these operators to some of the elements in*

$$\text{swap}_{(1,2)}((1,2,3)) = (2,1,3),\ \text{swap}_{(1,3)}((1,2,3)) = (1,2,3),$$
$$\text{swap}_{(1,2)}((1,3,2)) = (1,3,2),\ \text{swap}_{(1,3)}((1,3,2)) = (3,1,2),$$

*If the operators are applied to all* 24 *elements of $\mathcal{S}$ and recall the interpretation of the relation $R$ it can be see that* $(1,2,3),(2,1,3),(2,3,1)$ *are equivalent and* $(1,3,2),(3,1,2),(3,2,1)$ *are equivalent. Thus, we end up with the following two equivalent classes*

$$\mathcal{S}_1 = \{(1,2,3),(2,1,3),(2,3,1)\},$$
$$\mathcal{S}_2 = \{(1,3,2),(3,1,2),(3,2,1)\}, \quad (6)$$

*which forms a partition of the original set of crossing orders $\mathcal{S}$. Further, these equivalent classes translate to non-collision constraints of the type* (2b)

$$\mathcal{S}_1 \Rightarrow t_2(H_{23}) \leq t_3(L_{32}),\ \mathcal{S}_2 \Rightarrow t_3(H_{32}) \leq t_2(L_{23}).$$

*The sets* (6) *are identical to the grouping in Example 1 and thus the reduced number of crossing orders found in Example 2 contained exactly one element from each equivalent class.*

### *B. Algorithm provides minimum set*

In the previous section, we found a partition of the set of crossing orders $\mathcal{S}$. We denoted the sets of this partition by $[s_1], [s_2] \dots [s_k]$ and concluded that all elements of such a set yields the exact same collision constraints (2b). In this section, we prove that Algorithm IV.1 provides us with one and only one element from each of these sets.

**Theorem 2.** *Assume we have a k-dimensional crossing and $N$ vehicles, and at most one vehicle in each lane. Then, applying Algorithm IV.1 to the matrix of crossing orders $\mathcal{O}$ will result in a matrix of crossing orders $\mathcal{O}'$ which will contain exactly one crossing order from each equivalence class defined by the equivalence relation* (5)*.*

*Proof.* Applying the algorithm to the matrix of crossing orders $\mathcal{O}$ to obtain a reduced crossing order matrix $\mathcal{O}'$ is identical to applying it to the set of crossing orders $\mathcal{S}$ and end up with a subset $\mathcal{A}$ which contains the rows of $\mathcal{O}'$. This reduced set $\mathcal{A}$ is arrived at by removing one specific order of each redundant vehicle pair. Assuming one of these vehicle pairs is $(i,j)$ and all sequences containing that order is removed, then it means all elements $s \in \mathcal{S}$ have been removed for which $\text{swap}_{(i,j)}(s) = s_{j,i}$.
First, we prove that there is at least one element from each equivalent class in $\mathcal{A}$. This follows from transitivity of the equivalence relation in the following manner. Applying the algorithm to any element $s \in \mathcal{S}$ implies that

$$\text{swap}_1(s) = s_1, \text{swap}_2(s_1) = s_2, \dots, \text{swap}_n(s_{n-1}) = s_n,$$
$$\Rightarrow s R s_1, s_1 R s_2, \dots, s_{n-1} R s_n.$$

where the only element belonging to $\mathcal{A}$ is $s_n$, but by transitivity $s R s_n$ so $s_n$ is in the same equivalence class as $s$. Thus, all members of $\mathcal{S}$ are related by the equivalence relation $R$ to some element in $\mathcal{A}$.

Now, let us prove that $\mathcal{A}$ cannot contain two elements of any one equivalence class. Note that this is trivial if $\mathcal{A}$ contains only one element. Therefore assume that $\mathcal{A}$ contains at least two elements. For a proof of contradiction assume that the two elements $a_1, a_2 \in \mathcal{A}$ and that $a_1, a_2 \in [s]$. Then, we know from the construction of our equivalent classes that

$$a)\ \text{swap}_{(i,j)}(s) = a_1 \text{ or } b)\ \text{swap}^2_{(i,j)}(s) = a_1,$$
$$I)\ \text{swap}_{(k,l)}(s) = a_2 \text{ or } II)\ \text{swap}^2_{(k,l)}(s) = a_2,$$

for some redundant vehicle pairs $(i,j),(k,l) \in \mathcal{I}_1$. This leads us to four different cases:
b) and II): Since by definition the swap operator is its own inverse, it follows directly that $s = a_1 = a_2$. Contradiction.

a) and II): Then $s = a_2$ and $\text{swap}_{(i,j)}(s) = a_1$, but then $a_2$ is either equal to $a_1$ or $a_1$ and $a_2$ contain the sequence $(i,j)$ and $(j,i)$ respectively which means they cannot both belong to $\mathcal{A}$. Contradiction.
b) and I): Same argument as for a) and II).
a) and I): $s$ can be:

1) $s = s_{(i,j)}$ but not $(k,l)$ or $(l,k)$,
2) $s = s_{(j,i)}$ but not $(k,l)$ or $(l,k)$,
3) $s = s_{(i,j),(l,k)}$, i.e., booth,
4) $s = s_{(i,j,k)}$ where $j = l$.

If we have 1), 2) or 3) it is clear that one of $a_1$ and $a_2$ contain $(i,j)$ while the other contains $(j,i)$. Since $(i,j)$ is a redundant vehicle pair, only one of $a_1$ and $a_2$ can belong to $\mathcal{A}$. If we have 4) then $a_1$ contains $(j,i)$ and $a_2$ contains $(k,j)$. However, this would imply the sequences $(i,j)$ and $(k,j)$ has been removed, which is impossible since after the first of these were removed $j$ would be added to $\mathcal{I}_2$ which would prevent the removal of the other, by step 4 in Algorithm IV.1. So, only one of $a_1$ and $a_2$ can belong to $\mathcal{A}$. □

## VI. Case study

In this section the new formulation presented in this paper will be compared to the already existing formulation. The comparison will be made on the three scenarios depicted in Fig. 2-4. Each vehicle has three choices, move straight, turn right or turn left. In all three scenarios, the radius of a right turn is $2\,\mathrm{m}$, the radius of a left turn is $7.5\,\mathrm{m}$ while going straight implies infinite radius. The minimum velocity is set to $v_{\min} = 1\,\mathrm{km/h}$ and the legal speed limit $v_l = 90\,\mathrm{km/h}$. Further, the minimum and maximum acceleration of each vehicle in all scenarios is $a_{\min} = -4\,\mathrm{m/s^2}$ and $a_{\max} = 4\,\mathrm{m/s^2}$, respectively. All vehicles are assumed to start with acceleration 0 and a chosen reference velocity. The rest of the problem data is presented in Table I. Each quadratic optimization program is discreatized using first order Euler discretization with a sampling interval of $1\,\mathrm{m}$ and solved using the second order cone program solver ECOS, [16].

TABLE I
Problem data for Scenario 1, 2 and 3. Initial position, radius and acceleration limit is given for each vehicle.

| | Scenario 1 | Scenario 2 | Scenario 3 |
|---|---|---|---|
| Init. position (m) | [75, 70, 70] | [70, 70, 70, 70] | [70, 80, 80] |
| Radius (m) | $[2.5, \infty, \infty]$ | $[\infty, \infty, \infty, \infty]$ | [7.5, 7.5, 7.5] |
| $[w_{i1},\ w_{i2},\ w_{i3}]\ \forall i$ | [1, 1, 0.5] | [1, 1, 0.5] | [1, 1, 0.5] |

### A. Performance evaluation

Some results of solving the programs (1) and (2) for the Scenarios 1 to 3 are shown in Table II. The reference speeds, which are not constant due to the fact that the vehicles want to move slower when turning, are illustrated as dotted lines in the top plots of Figs. VI-A-5. As can be seen in the first row, the number of crossing orders that needed to be considered when using the intersection as a critical zone was 6, while it was reduced to 2 when using the shared zone approach (as expected from Examples 1 and 2). However, the crossing time has not been reduced in this case. This is because the reference velocity of the red vehicle is much larger than that of the green vehicle which allows them both to maintain their preferable speed in both cases. If, however, we increase the green vehicle's reference speed to $60\,\mathrm{km/h}$, changes in the acceleration of the green and red vehicles are made and using the shared zone approach the total crossing time is reduced by $0.4\,\mathrm{s}$.

Further, one can see from Table II that the total computation time for the QPs are lower for the shared zone approach compared to the intersection, and this is, in spite of, only two crossing orders providing feasible solutions in the case of intersections, i.e., only two of the six QPs needs to be solved. This is because the blue vehicle has a longitudinal velocity which is significantly lower than the red and green vehicles which makes it impossible for any of them to slow down enough to leave way for the blue vehicle, without violating the minimum acceleration constraints.

For the second scenario, which features four vehicles, one can most clearly see the benefits of the shared zone approach. The number of QPs needed to be solved have been reduced from 24 to 14 and the total optimization time has been reduced by $49\%$. The time to find the fourteen orders that needed to be solved took $51\,\mathrm{ms}$ which means that the total computation time was reduced by $45\%$. Further, the total time it took for the vehicles to travel through the crossing was reduced from $4.9\,\mathrm{s}$ to $2.9\,\mathrm{s}$.

In the third scenario, no reduction of the number of QPs solved is possible since all vehicles are crossing eachothers paths. Further, one can see very small variations in the solutions, the total crossing time has been reduced slightly but the calculation of the optimization problem has also gone up, slightly. Further, checking whether any crossing orders could be removed or not took an additional $14.28\,\mathrm{ms}$ which makes the shared zone approach $33\,\mathrm{ms}$ slower in this scenario. However, in an unpredictable traffic setting the small increase in computational time for simple scenarios such as this one might be worth it due to the large time reductions and improved performance possible in more complex scenarios as we saw in Scenario 2.

Lastly, we study the optimal speed and acceleration trajectories for all vehicles, which are presented in Fig. VI-A-5. It can be observed that the velocity and acceleration trajectories generated by the algorithm are smooth. Further, one can notice from the bottom plots of VI-A and 5 that the linearized acceleration constraints (dash-dotted lines) give rise to linearization error, since the optimal acceleration trajectories touch the linearized bounds. In practice, the linearization error could be removed by using sequential quadratic programming, [17].

## VII. Conclusions

In this paper we have provided an alternative formulation of the optimization based algorithm for safe and collaborative driving first introduced in [9] by introducing less restrictive collision constraints. This proposed approach decrease the number of quadratic optimization problems that need to

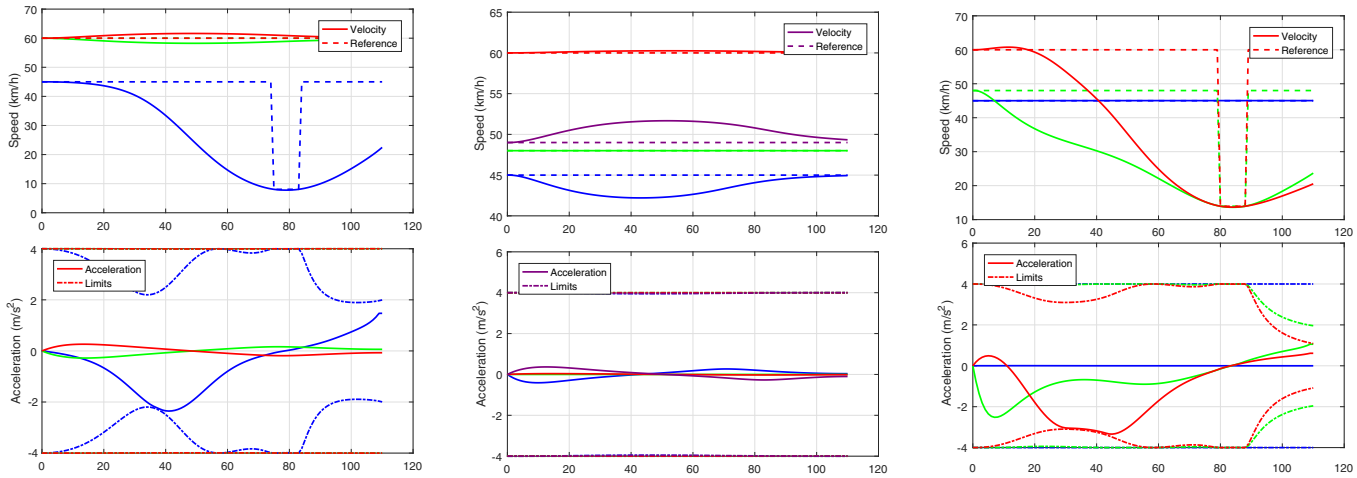


Fig. 5. Scenario 1-3 from left to right. Solid lines represent the optimal trajectories (speed in top figure, acceleration in the bottom figure), while dotted lines represent reference speed in the top plot and dash-dotted linearized acceleration limits in the bottom.

TABLE II

COMPARISON BETWEEN THE ALGORITHMS, ONE WITH THE CRITICAL ZONE BEING DEFINED BY THE INTERSECTION AND THE OTHER WHERE THE CRITICAL ZONES ARE DEFINED VIA SHARED ZONES. SCENARIOS, 1, 2 AND 3 REFER TO FIG. 2, 3 AND 4, RESPLECTIVELY.

| | Intersection | | | | Shared zones | | | |
|---|---|---|---|---|---|---|---|---|
| Scen. | Order | QP Calc. time (ms) | Tot. cross. time (s) | # Order | Order | QP Calc. time (ms) | Tot. cross time (s) | # Order |
| 1 | 3 2 1 | 128 | 11.15 | 6 | 3 2 1 | 69 | 11.14 | 2 |
| 2 | 3 4 2 1 | 1504 | 4.12 | 24 | 3 4 2 1 | 771 | 2.9 | 14 |
| 3 | 1 3 2 | 227 | 7.5 | 6 | 1 3 2 | 246 | 7.47 | 6 |

be solved to find the solution. An algorithm was formulated which was proven to provide the minimum number of optimization programs that needs to be solved to find the solution. Further, in simulation, it was seen that in scenarios with four vehicles the computation time could be significantly lowered.

## REFERENCES


[1] M. R. Hafner, D. Cunningham, L. Caminiti, and D. D. Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[2] S. Behere, M. Törngren, and D.-J. Chen, "A reference architecture for cooperative driving," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1095–1112, 2013.

[3] M. Simon, T. Hermitte, and Y. Page, "Intersection road accident causation: A european view," in *International Technical Conference on the Enhanced Safety of Vehicles*, 2009.

[4] "Crash factors in intersection-related crashes: An on-scene perspective," in *National Traffic Highway Safety Association*, 2010.

[5] A. Colombo and D. D. Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1515–1527, 2015.

[6] H. Ahn and D. D. Vecchio, "Safety verification and control for collision avoidance at road intersections," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 630–642, 2017.

[7] H. Kowshik, D. Caveney, and P. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, 2011.

[8] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artifical Intelligence*, vol. 31, no. 1, 2007.

[9] N. Murgovski, G. R. de Campos, and J. Sjöberg, "Convex modelling of conflict resolution at traffic intersections," in *Decision and Control (CDC)*, Osaka, Japan, 2015.

[10] L. Riegger, M. Carlander, N. Lidander, N. Murgovski, and J. Sjöberg, "Centralized mpc for autonomous intersection crossing," in *International Conference on Intelligent Transportation Systems*, 2016.

[11] A. Katriniok, P. Kleibaum, and M. Joševski, "Distributed model predictive control for intersection automation using a parallelized optimization approach," in *IFAC*, Toulouse, France, 2017.

[12] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using mode-based heuristics," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 8–21, 2017.

[13] L. Makarem and D. Gillet, "Model predictive coordination of autonomous vehicles crossing intersections," in *Preceedings of the 16th International IEEE Annual Conference on Intelligent Transporation Systems (ITSC)*, 2013.

[14] D. S. Dummit and R. M. Foote, *Abstract Algebra*, 3rd ed. John Wiley & Sons, 2004.

[15] J. R. Durbin, *Modern Algebra: An Introduction*, 6th ed. Wiley, 2005.

[16] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*, Zürich, Switzerland, 2013.

[17] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. Springer, 2000.