Transportation Research Part C 147 (2023) 103985

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Lane change scheduling for connected and autonomous vehicles

Maksat Atagoziev *, Ece Güran Schmidt, Klaus Werner Schmidt

Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, 06800, Turkey

ARTICLE INFO

Keywords:
Autonomous vehicles
Connected vehicles
Lane changes
Scheduling
Coordination
Cooperative adaptive cruise control
Real time

ABSTRACT

The execution of lane changes has a strong impact on driving safety and traffic throughput. Therefore, it is highly desired to automate and coordinate lane changes. The subject of this paper is the scheduling of lane changes of a group of connected and autonomous vehicles (CAVs) with the aim of minimizing the time during which all lane changes are completed, while keeping small inter-vehicle distances. To this end, the paper first develops an algorithm for minimizing the lane change time of a single CAV. This algorithm is then successively applied to all lane-changing CAVs. As a special feature, the proposed algorithm analytically computes CAV reference trajectories based on a second-order vehicle model such that all computations can be carried out in real time. In addition, the developed method supports the implementation of the computed CAV trajectories using cooperative adaptive cruise control in order to ensure safe driving in a tight vehicle formation. Detailed simulation experiments and a comparison to a benchmark method demonstrate the superior performance of our method.

1. Introduction

Connected and autonomous vehicles (CAVs) will populate our streets in the near future, offering advantages such as increased road safety, traffic throughput, fuel efficiency and driving comfort (Divakarla et al., 2019; Mariani et al., 2021; Malik et al., 2021). Hereby, it is expected that, different from performing the driving task as an individual agent, CAVs will cooperate with each other and with the road infrastructure via vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication to improve the situational awareness and to optimize traffic management (Luo et al., 2016; Martínez-Díaz and Soriguera, 2018; Eskandarian et al., 2019). In this context, lane changes (LCs) play an important role both regarding traffic safety and efficiency. On the one hand, uncoordinated LCs lead to perturbations on multilane freeways (Zheng et al., 2011) and hence increase travel time and fuel consumption (Barria and Thajchayapong, 2011). On the other hand, LCs are maneuvers that involve several vehicles and multiple lanes at the same time, which leads to a negative impact on traffic safety (Zheng, 2014). Referring to a recent survey, about 57% of the respondents report to have made sudden unsafe lane changes (Shawky, 2020). Accordingly, there is a strong motivation for the automation of LCs of CAVs (Malik et al., 2021).

There are different approaches for the automation of LCs. On the one hand, cooperative LCs assume that the ego-vehicle cooperates with its surrounding vehicles by exchanging local information in order to perform safe and efficient LCs (Nie et al., 2017; An and il Jung, 2018; Lin et al., 2019; Zheng et al., 2020; Wang et al., 2021). On the other hand, the coordination of CAV maneuvers and LCs is possible based on global information about the vehicle state in a certain road segment (Atagoziyev et al., 2016; Li et al., 2018, 2020; Chouhan et al., 2020; Xu et al., 2020). The main subject of this paper is the efficient computation of CAV trajectories for coordinated LCs when approaching critical points such as intersections, ramps or lane closures.

* Corresponding author.
E-mail addresses: e146329@metu.edu.tr (M. Atagoziev), eguran@metu.edu.tr (E. Güran Schmidt), schmidt@metu.edu.tr (K.W. Schmidt).

https://doi.org/10.1016/j.trc.2022.103985
Received 23 April 2022; Received in revised form 31 October 2022; Accepted 5 December 2022
Available online 26 December 2022
0968-090X/

When computing CAV trajectories for coordinated LCs in a group of CAVs that approach a critical point, several requirements need to be taken into account. From the traffic management perspective, it is desired to minimize LC times while enabling driving at small vehicle distances and with possible changes in the velocity of the group of CAVs. Considering traffic safety and driving comfort, it is necessary to compute vehicle trajectories that can be realized in practice and that adhere to pre-defined velocity and acceleration constraints. Specifically, trajectories for a robust realization should rather depend on the relative motion of vehicles with respect to each other than on the absolute vehicle position only. Finally, the actual implementation of algorithms for coordinated LCs is only possible if all computations can be carried out in real time.

There are several approaches in the existing literature that partially address the stated requirements. Li et al. (2018) and Cao et al. (2021) approach the LC coordination problem in two stages. Li et al. (2018) computes trajectories for all LC vehicles without accounting for collisions in the first stage. These trajectories are then shifted in order to obtain a sparse final configuration without overlap and an optimization problem is solved to compute a smooth transition between the initial configuration and the final configuration. In the second stage, all LC vehicles perform their LCs simultaneously. Cao et al. (2021) specifically focuses on merging at lane closures. In the first stage, an optimization problem is solved to open gaps between vehicles on normal lanes. A sorting algorithm then assigns vehicles from the blocked lane to these gaps for merging. Chouhan et al. (2020) divide the LC scenario into various independent frames under the assumption that all CAVs in a frame travel at a common velocity. A linear optimization problem is solved in each frame in order to compute channels to be used by LC vehicles while keeping a safe distance and minimizing the required deviation of all CAVs from their desired motion at the common velocity. Similarly, a given string of vehicles is divided into smaller groups by Li et al. (2020), whereby the trajectories of CAVs in successive groups are determined by the iterative solution of a nonlinear optimization problem. Lu et al. (2019) formulate the traffic management of vehicle trajectories as a mixed integer program, which is solved based on a rolling horizon algorithm in order to improve the computational efficiency. Hu and Sun (2019) and Xu et al. (2020) take into account the efficient LC computation at conflict zones such as merging areas. In Hu and Sun (2019), first a merging sequence of the vehicles in a group is specified based on the vehicle positions. Then, a linear programming model is developed to create sufficient gaps for LC vehicles. The recent work by Xu et al. (2020) proposes a general bi-level planning method based on the idea of cell discretization, where only one vehicle can occupy a cell. In the upper level, the right-of-way for vehicles is given based on a Monte Carlo tree search, whereas the actual trajectory planning and cost computation for each right-of-way decision is quickly resolved in the lower level.

It has to be noted that the methods by Li et al. (2018), Cao et al. (2021) and Chouhan et al. (2020) are only applicable under the assumption of a constant velocity of the considered group of vehicles. Furthermore, Chouhan et al. (2020), Li et al. (2020) and Lu et al. (2019) do not specifically address the fulfillment of real-time guarantees. Finally, it has to be pointed out that none of the discussed methods addresses the issue of a robust realization of CAV maneuvers based on relative positioning. This requirement is only taken into account in our previous work by Atagoziyev et al. (2016) but with a simple first-order dynamic vehicle model and without a detailed analysis and evaluation.

In this paper, we propose a new method for the LC scheduling and trajectory computation of CAVs at critical points such as intersections, ramps or lane closures with the aim of minimizing LC times while addressing all the previously stated requirements. Specifically, we first develop a LC algorithm that computes trajectories to minimize the LC time for a single CAV. We then apply this algorithm successively to all LC CAVs in order to identify the best positions for a minimum-time LC and compute the trajectories of LC CAVs and their surrounding CAVs accordingly. Hereby, the computation is based on analytical calculations, hence avoiding time-consuming optimization. In addition, all vehicle trajectories comply with the desired distance for cooperative adaptive cruise control (CACC) as in Darbha et al. (2020), ensuring a robust realization using relative vehicle positions. We illustrate the benefits of our method by several example scenarios and compare our results with a benchmark method.

The remainder of the paper is organized as follows. Section 2 introduces the vehicle and road segment model used in this paper and provides a formal problem statement. Section 3 first describes the main components of our method such as the efficient computation of connecting trajectories and minimum predecessor trajectories. These components are then used to develop our algorithm for a single lane change, which constitutes the basic building block of our novel algorithm for multiple lane changes in Section 4. A computational evaluation of our method is performed in Section 5 and conclusions and directions for future research are given in Section 6.

2. Model and problem formulation

This section provides our model for describing lane changes (LCs) of connected autonomous vehicles (CAVs) that approach a critical point on a road segment such as an intersection, ramp, or lane closure. Section 2.1 introduces the vehicle dynamics and Section 2.2 describes the underlying method for car following. The model of a road segment with a given number of CAVs is presented in Section 2.3 and the main problem addressed in this paper is formulated in Section 2.4.

2.1. Vehicle dynamics

We consider a set $\mathcal{V} = \{1, 2, \dots, n\}$ of n vehicles with a double integrator model similar to di Bernardo et al. (2015), Zhang and Orosz (2016) and Darbha et al. (2020). That is, writing x_i, v_i and a_i for the vehicle position, velocity and acceleration trajectories, respectively, the dynamic equations are given as

$$\begin{aligned} \dot{x}_i(t) &= v_i(t), \qquad x_i(0) = \hat{x}_{i,0} \\ \dot{v}_i(t) &= a_i(t), \qquad v_i(0) = \hat{v}_{i,0}. \end{aligned} \tag{1}$$

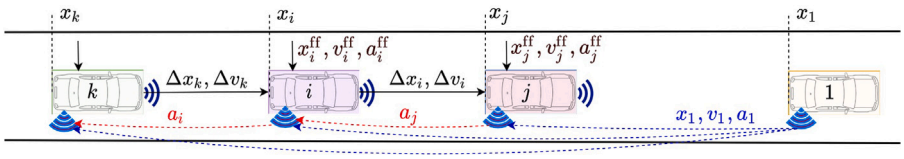

Fig. 1. CACC with additional feedforward signal for opening/closing gaps.

Hereby, a_i is the control input and $\begin{bmatrix} x_i & v_i \end{bmatrix}^T$ is the state vector of the vehicle with the initial position $\hat{x}_{i,0}$ and the initial velocity $\hat{v}_{i,0}$. Furthermore, we assume that velocity and acceleration are bounded such that for all times t,

$$v_{\min} \leq v_i(t) \leq v_{\max} \tag{2}$$

$$a_{\min} \leq \dot{v}_i(t) \leq a_{\max}. \tag{3}$$

In (2), $v_{\max}$ is the velocity limit on the respective road segment, whereas $v_{\min}$ characterizes the minimum velocity to be used during driving maneuvers. In general, $v_{\min}$ can be set to any value greater or equal to 0 m/s depending on the application scenario. In this paper, we choose $v_{\min}$ such that v_{nom} is equidistant from both $v_{\max}$ and $v_{\min}$. In addition, $a_{\max}$ and $a_{\min}$ in (3) represent acceleration limits that should be met for comfortable driving. Here, the literature suggests accelerations in the order of $a_{\min} = -2$ m/s^2 and $a_{\max} = 2$ m/s^2 (Wu et al., 2009; Matsuzaki et al., 2014; Nandi et al., 2015; Hoberock, 1977).

2.2. Car following

In this paper, we develop a method for scheduling LCs of CAVs. Such method is particularly useful in dense traffic, where CAVs drive in vehicle strings with small inter-vehicle distances, while allowing additional maneuvers such as opening/closing gaps before and after LCs.

In order to achieve this task, we propose a setting, where CAVs apply cooperative adaptive cruise control (CACC) to realize tight car following using the constant gap policy that is for example employed by Darbha et al. (2020). That is, as illustrated in Fig. 1, we consider that each vehicle in a vehicle string follows its predecessor vehicle at a constant distance. For example, vehicle i with its predecessor vehicle j in Fig. 1 has the desired gap distance d_i. The desired position of vehicle i is hence computed as

$$x_i(t) = x_j(t) - d_i. \tag{4}$$

When applying CACC with the constant gap policy in (4), it is common to use leader-predecessor following (LPF) as the communication topology. As can be seen in Fig. 1, each vehicle i uses local measurements (for example from a Radar sensor) of the distance Δx_i and the velocity difference Δv_i to the direct predecessor vehicle j. In addition, the predecessor vehicle j communicates its acceleration a_j, whereas the leader vehicle 1 communicates its position x_1, velocity v_1 and acceleration a_1. Vehicle i then uses this feedback (fb) information to compute its acceleration input a_i^{fb}.

When using CACC as described above, it is the case that any vehicle i follows its predecessor at a constant distance d_i according to (4). Nevertheless, it is possible to realize additional maneuvers such as opening or closing gaps between vehicles by a modified version of CACC that was introduced by Sağlam and Schmidt (2018). Consider that for example vehicle i wants to open a gap to its predecessor vehicle j in Fig. 1. Then, additional feedforward (ff) signals a_i^{ff} and x_i^{ff} for the acceleration and position of vehicle i are applied such that vehicle i tracks the modified desired position

$$x_i(t) = x_j(t) - d_i + x_i^{\mathrm{ff}}(t) \tag{5}$$

using the modified acceleration input

$$a_i(t) = a_i^{\mathrm{fb}}(t) + a_i^{\mathrm{ff}}(t), \tag{6}$$

whereby $a_i^{\mathrm{ff}}(t) = \ddot{x}_i^{\mathrm{ff}}(t)$. Note that the classical constant gap policy in (4) is achieved if a_i^{ff} and x_i^{ff} are zero.

We next provide an example with 6 vehicles for illustration. Here, the CACC method in Darbha et al. (2020) is extended by the described modification in (5) and (6). In the example, the leader vehicle 1 performs a speed-up maneuver between time $t = 1$ s and $t = 3$ s, whereas the remaining vehicles apply car-following with a constant gap of 20 m. In addition, vehicle 3 and 5 open gaps of 20 m between time $t = 4$ s and $t = 10$ s, and time $t = 12$ s and $t = 18$ s, respectively. The applied feedforward accelerations a_3^{ff}, a_5^{ff} and the corresponding feedforward positions $x_3^{\mathrm{ff}}, x_5^{\mathrm{ff}}$ are shown in Fig. 2.

Furthermore, Fig. 3 displays the resulting position and velocity trajectories of all vehicles. Specifically, it can be seen that all vehicles apply the same velocity increase between time $t = 1$ s and $t = 3$ s. In addition, it holds that all vehicles successfully apply car following at a constant distance even while other vehicles open a gap. The zoomed-in part of the right-hand plot further illustrates the application of CACC with the additional gap-opening maneuver in the sense that the computed feedforward signals serve as reference signals that are tracked by the feedback loop of CACC.

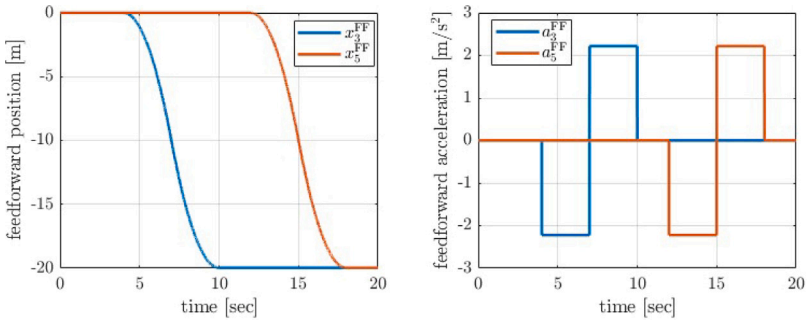

Fig. 2. Feedforward signals of vehicle 3 and 5.

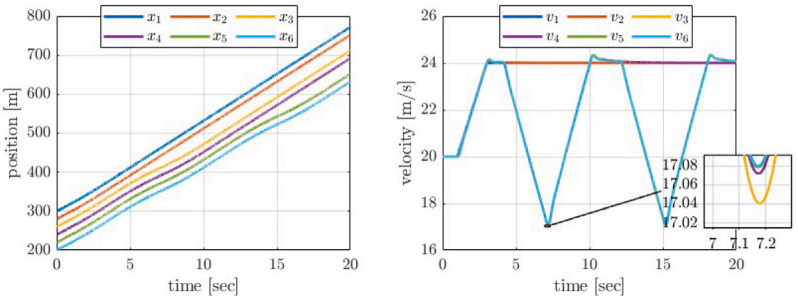

Fig. 3. Vehicle trajectories when applying CACC with additional feedforward signals.

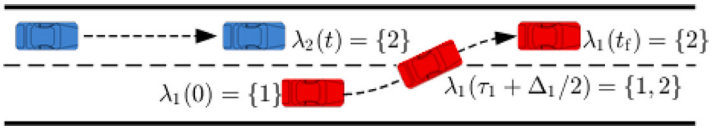

Fig. 4. Lane occupation of CAVs.

Remark 1. We note that existing work such as Li et al. (2018) assumes that vehicles individually track absolute position trajectories without taking into account the relation between vehicles.

Differently, the position trajectory in (5) is defined relative to the desired position for car following and can hence be realized without the need for a highly precise absolute vehicle position. This both improves the robustness of the implementation and increases driving safety.

2.3. Road segment model

We assume that the vehicles $\mathcal{V}$ under consideration travel on a road segment with two lanes (numbered 1 and 2) for a certain amount of time t_f. Here, t_f represents the maximum admissible time for a maneuver of the vehicles in $\mathcal{V}$ such as reaching an intersection, an on/off ramp or a lane closure. We write $\lambda_i(t)$ for the lane of vehicle i at time t and introduce $\lambda_{i,0} \in \{1, 2\}$ and $\lambda_{i,f} \in \{1, 2\}$ as the initial and final lane, respectively. That is, $\lambda_i(0) = \lambda_{i,0}$ and $\lambda_i(t_f) = \lambda_{i,f}$. We characterize a LC of vehicle i by the time τ_i, where the LC starts and the duration Δ_i of the LC similar to Hu and Sun (2019). In particular, such LC occurs in the time interval $[\tau_i, \tau_i + \Delta_i]$ and vehicle i occupies both lanes during that time. Accordingly, the lane of vehicle i is characterized by

$$\lambda_i(t) = \begin{cases} \{\lambda_{i,0}\} & \text{for } t < \tau_i \\ \{\lambda_{i,0}\} \cup \{\lambda_{i,f}\} & \text{for } \tau_i \le t \le \tau_i + \Delta_i \\ \{\lambda_{i,f}\} & \text{otherwise} \end{cases} \tag{7}$$

Hereby, we note that this definition also captures vehicles that do not change the lane by assigning $\tau_i = 0$ for these vehicles. An illustration of (7) is given in Fig. 4 for an LC vehicle 1 and a non-LC vehicle 2.

We next introduce the notion of a path of vehicle i as a tuple $\mathcal{P}_i = (x_i, v_i, \lambda_i)$, whereby the position trajectory x_i and velocity trajectory v_i fulfill (1) to (3) with $x_i(0) = \hat{x}_{i,0}$ and $v_i(0) = \dot{x}_i(0) = \hat{v}_{i,0}$ and λ_i fulfills (7). Considering all vehicles in $\mathcal{V}$, a group path is defined by the set $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ such that $\mathcal{P}_i$ is a vehicle path for each $i \in \mathcal{V}$. In principle, it is desired that all vehicles maintain

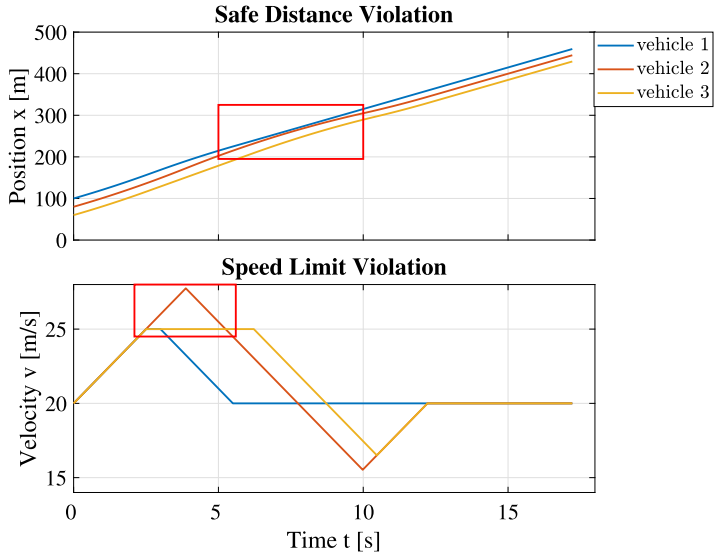

Fig. 5. Infeasible group path illustration: Safe distance violation (top); maximum velocity violation (bottom).

a minimum distance as given in (4) for driving safety. Accordingly, we denote a group path $\mathcal{P}$ as feasible if it holds that $\mathcal{P}_i$ is a vehicle path for all $i \in \mathcal{V}$ and $\forall i, j, i \neq j$ and $\forall t$, $0 \leq t \leq t_f$,

$$\lambda_i(t) \cap \lambda_j(t) \neq \emptyset \wedge x_i(t) > x_j(t) \Rightarrow x_i(t) - x_j(t) \geq d_j \tag{8}$$

That is, all vehicles that occupy the same lane at a certain time t should keep a minimum safety gap of d_j. Consider for example Fig. 5, assuming a safe distance of $d_i = 15\,\text{m}$ for all vehicles $i = 1, 2, 3$ and a maximum velocity of $v_{\max} = 25\,\text{m/s}$. Here, vehicle 2 comes too close to vehicle 1, whereas vehicle 3 keeps a safe distance to vehicle 2. In addition, vehicle 2 exceeds the maximum velocity. Together, the group path $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ is infeasible.

We further write $\mathcal{F}$ for the set of all feasible group paths and define the LC time $\tau_{\mathcal{P}}$ of a feasible group path $\mathcal{P} \in \mathcal{F}$ as the time where the last LC is completed:

$$\tau_{\mathcal{P}} = \max_{i \in \mathcal{V}} \left(\tau_i + \Delta_i\right). \tag{9}$$

2.4. Problem formulation

In this paper, we are interested in finding feasible group paths for a group of vehicles $\mathcal{V}$ while minimizing the time for completing all LCs while approaching a critical point on a road segment. This goal is achieved by finding a feasible group path that minimizes the maximum LC time in (9) as follows.

$$\min_{\mathcal{P} \in \mathcal{F}} \left\{\tau_{\mathcal{P}}\right\}. \tag{10}$$

In addition, we assume that the group of vehicles is supposed to exhibit a joint behavior (such as following a specified speed profile) that is determined by a traffic management system. To this end, we introduce a virtual leader path $\mathcal{P}_0$ that specifies the desired joint behavior of the vehicles in $\mathcal{V}$ when traveling towards the critical point. Specifically, it is desired that the positions $x_i(t)$ of all vehicles $i \in \mathcal{V}$ stay as close as possible to the position $x_0(t)$ of the virtual leader in order to obtain a tight formation when reaching the critical point. That is, we want to minimize the maximum distance from the virtual leader as

$$\min_{\mathcal{P} \in \mathcal{F}} \left\{ \max_{i \in \mathcal{V}, 0 \leq t \leq t_f} \{|x_0(t) - x_i(t)|\} \right\}. \tag{11}$$

Hereby, we note that minimizing the distance of the last vehicle in the group to the leader ensures that also the remaining vehicles stay close to the virtual leader.

Using the notation introduced before, it is now possible to state the problem addressed in this paper. Given a set of vehicles $\mathcal{V}$ with their respective initial positions $\hat{x}_{i,0}$, velocities $\hat{v}_{i,0}$ and lanes $\lambda_{i,0}$ as well as final lanes $\lambda_{i,f}$, and introducing the virtual leader path $\mathcal{P}_0$, we want to solve the following joint optimization problem:

$$\min_{\mathcal{P} \in \mathcal{F}} \left\{ \tau_{\mathcal{P}}, \max_{i \in \mathcal{V}, 0 \leq t \leq t_f} \{|x_0(t) - x_i(t)|\} \right\}. \tag{12}$$

That is, we want to find a feasible group path $\mathcal{P} \in \mathcal{F}$ such that the overall LC completion time $\tau_{\mathcal{P}}$ in (9) is minimal, while all vehicles in $\mathcal{V}$ stay as close as possible to the virtual leader position x_0.

We note that the described problem is a computationally hard problem. Since there is no indication of the optimal order of CAVs on each lane after all LCs are completed, it is in principle required to consider each possible vehicle order. Then, for each vehicle order, a minimum-time optimal control problem with a large number of states (depending on the number of CAVs) and time-varying constraints (given by the minimum safety distance) has to be solved.

In this paper, we tackle the stated problem by a heuristic approach with the following features.

- F1 Since CAVs travel in a given direction, LCs of CAVs that are more ahead affect the motion of CAVs behind. Hence, we step-by-step compute the paths of CAVs starting from CAVs in the front.
- F2 Since the computation of vehicle paths must be carried out in real time, we avoid solving optimization problems but rather favor analytical trajectory computations.
- F3 We compute vehicle trajectories based on the desired distance for CACC defined in (4).
- F4 We do not put any restriction on the LC time of individual LC CAVs as long as the overall LC completion time is minimized.
- F5 Our method is not restricted to the case where the leader vehicle travels at a constant speed.

We note that a heuristic solution based on the above assumptions is very likely not optimal. However, noting that no optimal solution to the described LC scheduling problem exists in the literature, the method proposed in this paper has the following important advantages. First, F1 ensures that only a limited number of trajectory computations is required. Second, F2 removes the need for optimization such that solution trajectories can be found in real time. Third, different from all the existing work, F3 enables the direct use of the computed trajectories in practice as described in Section 2.2. Fourth, F4 allows LCs at arbitrary times before the overall LC completion time, which is different from existing work such as Li et al. (2018), where all LCs need to happen at the same time. In other words, our formulation allows vehicles to switch to their target lane as soon as the minimum required space is opened. Finally, F5 allows for applications with variable speeds of a group of CAVs different from Li et al. (2018) and Chouhan et al. (2020). Accordingly, our LC scheduling algorithm can be used as a building block for advanced ITS applications such as scheduling groups of vehicles at an intersection or merging at a lane closure.

3. Single lane change algorithm

We next address the LC computation for a single vehicle. To this end, Section 3.1 first identifies a suitable class of trajectories. Then, Sections 3.2 and 3.3 present conditions and methods for connecting different trajectories during or after LC maneuvers that can be evaluated efficiently. Finally, Section 3.4 proposes our algorithm for a single LC, which serves as a building block for our general LC method in Section 4.

3.1. Position and velocity trajectories

In general, vehicles need to adjust their speed to arrange gaps for LC maneuvers in compliance with the safety distance in (4). Considering the vehicle model in (1), this requires supplying appropriate acceleration inputs for each vehicle $i \in \mathcal{V}$, while respecting the constraints in (2) and (3). Since we are interested in fast LC maneuvers, we propose to use vehicle trajectories that are generated by the possible input signals $a_i(t) = a_{\max}$ (maximum acceleration), $a_i(t) = a_{\min}$ (minimum acceleration) and $a_i(t) = 0$ (constant speed).

In line with the previous discussion, we use piece-wise constant acceleration inputs such that a position trajectory x_i of vehicle i is a concatenation of $m_i + 1$ trajectory pieces, where the position at time t is of the form

$$x_{i,k}(t) = \hat{x}_{i,k} + \hat{v}_{i,k} \cdot (t - t_{i,k}) + \frac{\hat{a}_{i,k}}{2} \cdot (t - t_{i,k})^2, \quad k = 0, \dots, m_i. \tag{13}$$

Hereby, we assume that the kth trajectory piece of vehicle i is valid in the interval $[t_{i,k}, t_{i,k} + \Delta_{i,k}]$ such that $t_{i,k}$ is the start time and $\Delta_{i,k}$ is the duration. Moreover, $\hat{x}_{i,k}$ and $\hat{v}_{i,k}$ denote the position and velocity values at time $t_{i,k}$, whereas $\hat{a}_{i,k} \in \{a_{\min}, 0, a_{\max}\}$ represents the constant acceleration of the trajectory piece. That is, each trajectory piece is fully described by the tuple $(t_{i,k}, \Delta_{i,k}, \hat{x}_{i,k}, \hat{v}_{i,k}, \hat{a}_{i,k})$. In addition, it must hold for consecutive trajectory pieces that

$$t_{i,k+1} = t_{i,k} + \Delta_{i,k}, \hat{x}_{i,k+1} = x_{i,k}(t_{i,k+1}), \hat{v}_{i,k+1} = v_{i,k}(t_{i,k+1})$$

Considering that trajectory pieces are either parabolic or linear, we write P^+, P^- and L for trajectory pieces with $a_{i,k} = a_{\max}$, $a_{i,k} = a_{\min}$ and $a_{i,k} = 0$, respectively.

3.2. Connecting trajectories

The LC algorithm developed in Section 3.4 is based on different types of connecting trajectories that are required for opening/closing gaps as well as smoothing discontinuities because of vehicles that enter or leave lanes. In this context, we use the term connecting trajectory to represent curves used to smoothly connect two distinct trajectories and hence avoid discontinuities. The main aim of using such connecting trajectories is to maintain drivable trajectories that can serve as reference trajectories for the CACC model in Section 2.2 in each step of the algorithm.

We first consider forward connections, where it is desired to smoothly connect a point on a path $\mathcal{P}_0$ to some point on another path $\mathcal{P}_i$. Formally, the point can be characterized by a tuple $(t_0, x_0(t_0), v_0(t_0))$, where $(x_0(t_0), v_0(t_0))$ is a position/velocity pair at a

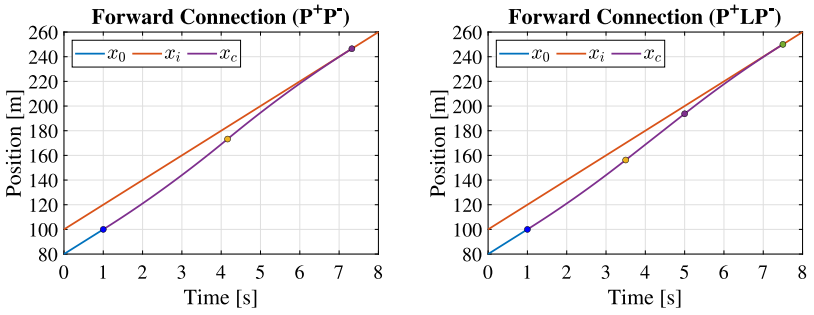

Fig. 6. Forward connections: P^+P^- connection (left); P^+LP^- connection (right).

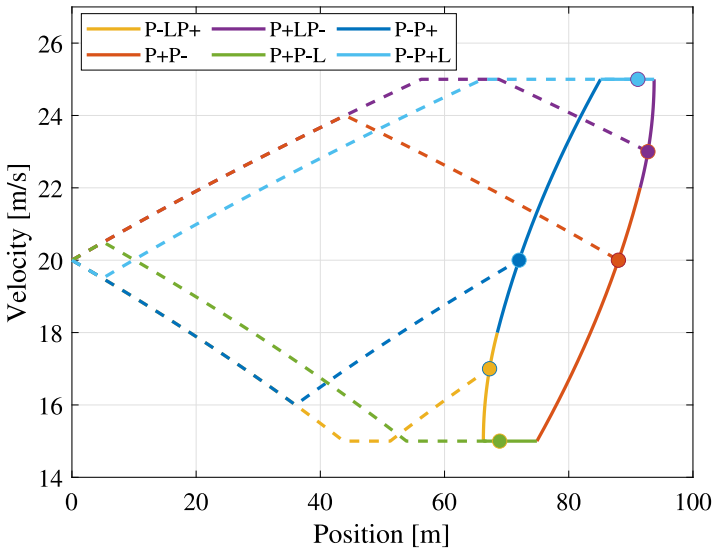

Fig. 7. Illustration of the reachable set after time $t - t_0 = 5$ s.

certain time t_0. Then, it is desired to find a path $\mathcal{P}_{\text{FW}}$ that follows $\mathcal{P}_0$ until t_0, then smoothly connects to $\mathcal{P}_i$ at the minimum possible time t_c and follows $\mathcal{P}_i$ afterwards.

Illustrations for such forward connection are shown in Fig. 6, where $t_0 = 1$ s. Forward connections are for example needed when opening or closing a gap between vehicles. In this case, x_0 corresponds to the position trajectory of a vehicle that has a certain distance to its desired trajectory x_i at time t_0 and it is desired to connect from $x_0(t_0)$ to the desired position trajectory x_i in a minimal time.

Similar to the work by Johnson and Hauser (2012) and Nguyen and Au (2019), we suggest to concatenate P^+, P^- and L-trajectory pieces as defined in Section 3.1 in order to obtain forward connections. For example, we write P^+P^- for trajectories that start with a P^+ trajectory piece that is followed by a P^- trajectory piece. Overall, we use P^+P^-, P^-P^+, P^+LP^-, P^-LP^+, P^+P^-L and P^-P^+L trajectories. Specifically, it holds that the set of reachable states (position-velocity pairs) of such trajectories from a given point $(t_0, x_0(t_0), v_0(t_0))$ at any time $t \geq t_0$ is described by a closed curve in the position-velocity space as shown in Fig. 7. Hereby, the upper and lower straight boundaries exist if $v_{\max}$ and $v_{\min}$ can be reached from $v_0(t_0)$ within time $t - t_0$. In the figure, $(t_0, x_0(t_0), v_0(t_0)) = (0, 0, 20)$, $t = 5$ s, $v_{\max} = 25$ m/s and $v_{\min} = 15$ m/s.

In the sequel, we write $\mathcal{R}(x_0(t_0), v_0(t_0), t - t_0)$ for the set of reachable states from $(x_0(t_0), v_0(t_0))$ at time t and $\overline{\mathcal{R}}(x_0(t_0), v_0(t_0), t - t_0)$ for the set of states enclosed by $\mathcal{R}(x_0(t_0), v_0(t_0), t - t_0)$.

As the first important fact, we conclude that a smooth connection from $(t_0, x_0(t_0), v_0(t_0))$ to $\mathcal{P}_i$ exists if and only if there is a time $t_c \geq t_0$ such that $(x_i(t_c), v_i(t_c)) \in \mathcal{R}(x_0(t_0), v_0(t_0), t_c - t_0)$. In that case, there must be a connecting trajectory denoted by x_c and v_c such that

$$x_c(t_0) = x_0(t_0) \text{ and } x_c(t_c) = x_i(t_c)$$
$$v_c(t_0) = v_0(t_0) \text{ and } v_c(t_c) = v_i(t_c)$$

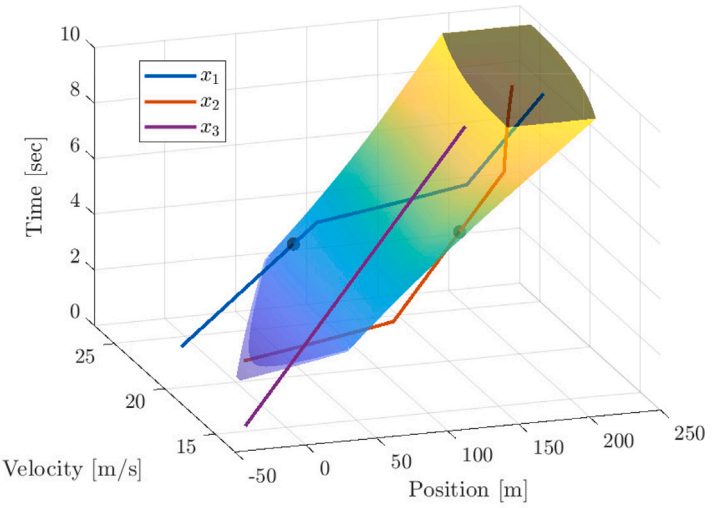

Fig. 8. Illustration of the reachable set until time $\Delta t = 10\,\mathrm{s}$.

Accordingly, the concatenated trajectory

$$\begin{cases} x_{\mathrm{FW}}(t) = x_0(t), v_{\mathrm{FW}}(t) = v_0(t) & \text{for } t \le t_0 \\ x_{\mathrm{FW}}(t) = x_{\mathrm{c}}(t), v_{\mathrm{FW}}(t) = v_{\mathrm{c}}(t) & \text{for } t_0 < t \le t_{\mathrm{c}} \\ x_{\mathrm{FW}}(t) = x_i(t), v_{\mathrm{FW}}(t) = v_i(t) & \text{otherwise} \end{cases} \tag{14}$$

is also a feasible trajectory. This fact is illustrated for the example of $\mathrm{P^+P^-}$ and $\mathrm{P^+LP^-}$ forward connections in Fig. 6.

In addition, it is possible to analytically decide the existence of such smooth connection before a given time $\hat{t}$. In particular, there must exist a smooth connecting trajectory with x_{c} and v_{c} for some $t_{\mathrm{c}} \le \hat{t}$ if and only if $(x_i(\hat{t}), v_i(\hat{t})) \in \overline{\mathcal{R}}(x_0(t_0), v_0(t_0), \hat{t} - t_0)$. The reason is that, if $(x_i(\hat{t}), v_i(\hat{t})) \in \overline{\mathcal{R}}(x_0(t_0), v_0(t_0), \hat{t} - t_0)$, it must hold that $(x_i(t), v_i(t))$ intersects the boundary curve $\mathcal{R}(x_0(t_0), v_0(t_0), t_{\mathrm{c}} - t_0)$ for some $t_{\mathrm{c}} \le \hat{t}$.

This fact is also illustrated in Fig. 8, which shows three example trajectories $(x_1(t), v_1(t))$, $(x_2(t), v_2(t))$ and $(x_3(t), v_3(t))$. Here, $t_0 = 0$, $x_0(t_0) = 0$, $v_0(t_0) = 20\,\mathrm{m/s}$ and $\hat{t} = 10\,\mathrm{s}$. It is readily observed that $(x_1(\hat{t}), v_1(\hat{t}))$ and $(x_2(\hat{t}), v_2(\hat{t}))$ are in $\overline{\mathcal{R}}(x_0(t_0), v_0(t_0), \hat{t} - t_0)$. Accordingly, there is a connection from $(x_0(t_0), v_0(t_0))$ to both $(x_1(t), v_1(t))$ and $(x_2(t), v_2(t))$ in Fig. 8, which is indicated by the points in the figure. Differently, $(x_3(\hat{t}), v_3(\hat{t})) \notin \overline{\mathcal{R}}(x_0(t_0), v_0(t_0), \hat{t} - t_0)$ such that there is no connection from $(x_0(t_0), v_0(t_0))$ to $(x_3(t), v_3(t))$ within time $\hat{t}$.

Using the previous facts, it is now possible to define a procedure for computing the described smooth connection from a given point $(x_0(t_0), v_0(t_0))$ on path $\mathcal{P}_0$ to a generic path $\mathcal{P}_i$ with m_i trajectory pieces.

Algorithm 1 Algorithm for forward connection.

$f_{\mathrm{FW}}(t_0, \mathcal{P}_0, \mathcal{P}_i, t_i, a)$
Output: $\mathcal{P}_{\mathrm{FW}}$, t_{FW}
1: **if** $(x_i(t_i), v_i(t_i)) \in \overline{\mathcal{R}}(x_0(t_0), v_0(t_0), t_i - t_0)$ **then**
2: **for** $k = 0, \ldots, m_i$ **do**
3: Compute connecting trajectory x_{c}, v_{c} and t_{c} to $x_{i,k}$ (among $\mathrm{P^+P^-}$, $\mathrm{P^-P^+}$, $\mathrm{P^+LP^-}$, $\mathrm{P^-LP^+}$)
4: **if** $t_{i,k} \le t_{\mathrm{c}} \le t_{i,k} + \Delta_{i,k}$ **then**
5: **break**
6: Define x_{FW} and v_{FW} as in (14), $t_{\mathrm{FW}} = t_{\mathrm{c}}$
7: **else**
8: **for** $t \in [t_0, t_{\mathrm{f}}]$ **do**
9: $x_{\mathrm{FW}}(t) = x_0(t_0) + v_0(t_0)\,(t - t_0) + \frac{a}{2}\,(t - t_0)^2$
10: $v_{\mathrm{FW}}(t) = v_0(t_0) + a\,(t - t_0)$ and $t_{\mathrm{FW}} = \infty$
11: $\lambda_{\mathrm{FW}}(t) = \lambda_0(t_0), \forall t$
12: **return** $\mathcal{P}_{\mathrm{FW}}$ and t_{FW}

Algorithm 1 first checks if a smooth connection exists (line 1 in Algorithm 1). In the positive case, a connection is attempted for each trajectory piece $x_{i,k}$ of x_i. If such connection is found within the time interval $[t_{i,k}, t_{i,k} + \Delta_{i,k}]$, where $x_{i,k}$ is valid, the connection is accepted and the overall path is determined in line 6 and 11 in Algorithm 1. Hereby, we know from the discussion above that such connection must exist for some $x_{i,k}$, $0 \le k \le m_i$ among $\mathrm{P^+P^-}$, $\mathrm{P^-P^+}$, $\mathrm{P^+LP^-}$ or $\mathrm{P^-LP^+}$ trajectories if line 1 in Algorithm 1 evaluates to true. If no connection exists, the algorithm computes a trajectory with a specified acceleration a (line 9 and 10 in Algorithm

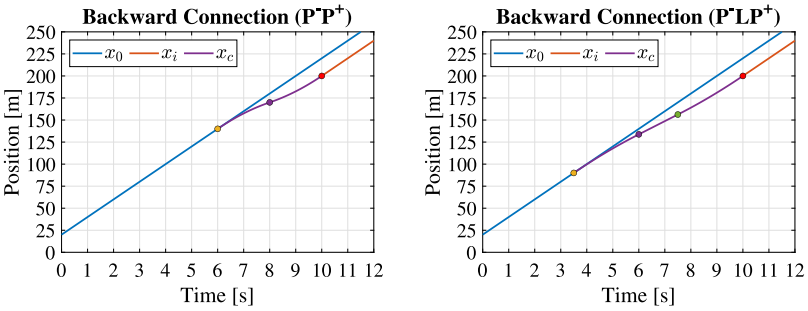

Fig. 9. Backward connections: P^-P^+ (left); P^-LP^+ (right).

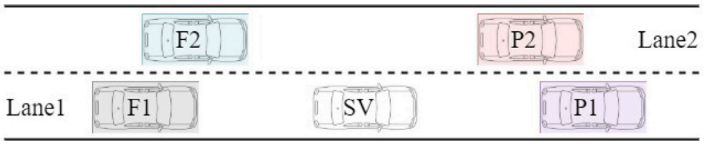

Fig. 10. Vehicles involved in a LC.

1) and returns $t_{\mathrm{FW}} = \infty$ since no connection was found. That is, in any case, Algorithm 1 returns a valid path $\mathcal{P}_{\mathrm{FW}}$. Regarding the computational complexity, determining the connection only requires the solution of a quadratic equation, which can be done in constant time. Overall, the complexity of Algorithm 1 is determined by the number of iterations of the for loop in line 2 and hence $O(m_i)$. We next consider the computation of a backward connection. Here, it is desired to find a smooth connection from a path $\mathcal{P}_0$ to a given point $(t_i, x_i(t_i), v_i(t_i))$ on another path $\mathcal{P}_i$. Such connection is for example needed when a gap has to be opened for a LC vehicle until a certain time t_i as illustrated in Fig. 9. Here, a CAV should follow the position trajectory x_0 before and x_i after the initiation of a LC at time $t_i = 10\,\mathrm{s}$. That is, the CAV has to start opening a gap sufficiently before $t_i = 10\,\mathrm{s}$, which can be computed by finding a backward connection from the point $(t_i, x_i(t_i), v_i(t_i))$ to the path $\mathcal{P}_0$. Since the computation of the backward reachable set is analogous to the forward reachable set described above, we do not provide a detailed description. The corresponding function is denoted as $f_{\mathrm{BW}}(t_0, \mathcal{P}_0, \mathcal{P}_i, t_i, a)$, whereby t_0 denotes the earliest allowed time for the connection.

3.3. Minimum predecessor trajectory

The connections computed in the previous section are concerned with vehicle trajectories on a single lane. In addition, it has to be taken into account that a LC vehicle has to keep a safe distance to its surrounding vehicles. To this end, the vehicles involved in a LC of a given LC CAV, denoted as subject vehicle (SV), are depicted in Fig. 10. In general, the SV has predecessor and follower vehicles on both lanes, which are denoted as P1, P2, F1 and F2 with their respective paths $\mathcal{P}_{\mathrm{P1}}$, $\mathcal{P}_{\mathrm{P2}}$, $\mathcal{P}_{\mathrm{F1}}$ and $\mathcal{P}_{\mathrm{F2}}$. Considering the roles of these vehicles for a LC of the SV, we denote the predecessor on the current lane and the predecessor on the target lane as current predecessor (CP) and target predecessor (TP), respectively. For example, if the SV is located on lane 1 (as in Fig. 10), it holds that CP=P1 and TP=P2. Finally, the target follower (TF) vehicle is the follower vehicle on the target lane. In Fig. 10, TF=F2.

Based on the above definitions, it is the case that the SV can perform a LC if its position is at a safe distance to the CP, TP and TF. During a LC, it further has to be taken into account that the CP and TP can overtake each other as can be seen in Fig. 11. Hence, in principle, the closest of these vehicles determines the desired position of the SV. To this end, we define

$$\hat{x}_{\mathrm{MP}}(t) = \min\{x_{\mathrm{CP}}(t), x_{\mathrm{TP}}(t)\}, \forall t \tag{15}$$

as the minimum predecessor (MP) position. In addition, we introduce $\hat{v}_{\mathrm{MP}}(t)$ as the corresponding velocity values of the respective vehicle with the minimum position at time t.

Considering $\hat{x}_{\mathrm{MP}}$ and $\hat{v}_{\mathrm{MP}}$ as defined before, it has to be noted that the velocity trajectory $\hat{v}_{\mathrm{MP}}$ is not continuous and hence cannot be followed by the SV during a LC according to the vehicle model in (1). For example, in Fig. 11, the CP overtakes the TP at time $t = 3.8\,\mathrm{s}$. That is, until this time, the SV should follow the CP with its velocity of $25\,\mathrm{m/s}$. However, after $t = 3.8\,\mathrm{s}$, the SV should follow the (closer) TP with a velocity of $15\,\mathrm{m/s}$. This requires an instantaneous jump in the velocity, which is clearly infeasible. Because of this reason, we propose Algorithm 2 that iteratively refines $\hat{x}_{\mathrm{MP}}$ and $\hat{v}_{\mathrm{MP}}$ until a continuously differentiable MP trajectory x_{MP} is obtained.

The algorithm starts from the computation of the initial MP trajectory in line 1 in Algorithm 2, where it is known that the trajectory is continuously differentiable until time $t_{\mathrm{C}} = 0$. In each iteration of the algorithm, the next discontinuity after t_{C} is determined (line 3 in Algorithm 2). If there is no more discontinuity, the algorithm terminates (line 5 in Algorithm 2). Otherwise, there is a discontinuity at t_{D}. The algorithm considers all trajectory pieces of x_{MP} before t_{D} and tries to find a connecting trajectory

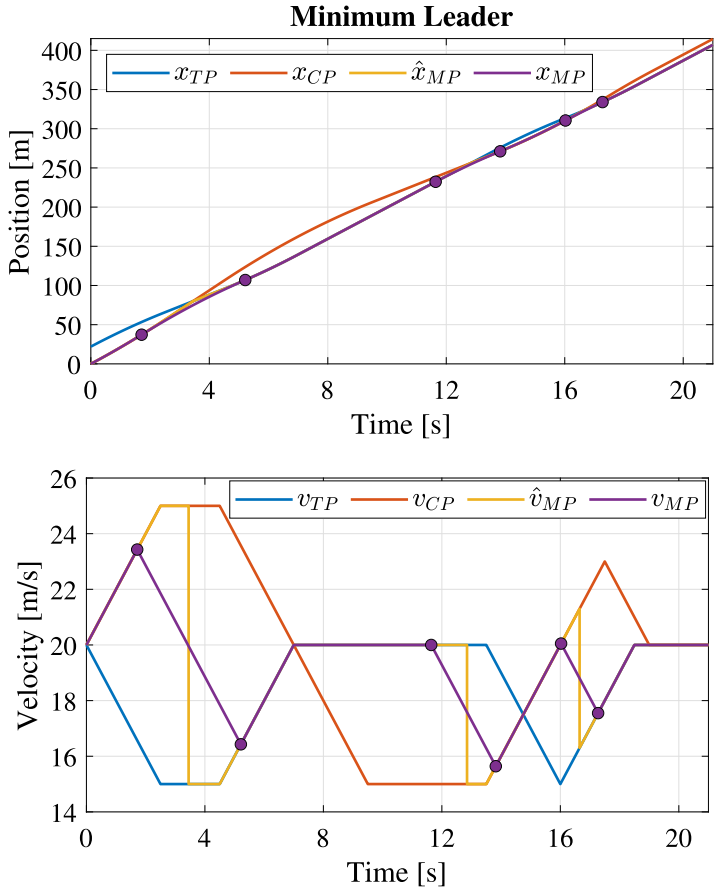

Fig. 11. Minimum predecessor computation.

Algorithm 2 Minimum predecessor computation.

$f_{\text{MP}}(\mathcal{P}_{\text{CP}}, \mathcal{P}_{\text{TP}})$
Output: $\mathcal{P}_{\text{MP}}$
1: **Initialize:** $x_{\text{MP}} = \hat{x}_{\text{MP}}$ and $v_{\text{MP}} = \hat{v}_{\text{MP}}$; set $t_{\text{C}} = 0$
2: **while true do**
3: Determine the next discontinuity in v_{MP} at $t_{\text{D}} \geq t_{\text{C}}$
4: **if** No discontinuity is found **then**
5: **break**
6: Let $l = \arg\max_k \{t_{\text{MP},k} \leq t_{\text{D}}\}$
7: **for** $k = l, l-1, \ldots, 0$ **do**
8: $(\hat{\mathcal{P}}_{\text{MP}}, t_{\text{c}}) = f_{\text{FW}}(t_{\text{MP},k}, \mathcal{P}_{\text{MP}}, \mathcal{P}_{\text{MP}}, t_{\text{f}}, a_{\min})$
9: **if** $t_{\text{c}} < t_{\text{D}}$ **or** x_{c} intersects x_{MP} before t_{c} **then**
10: **continue**
11: **else**
12: **if** $t_{\text{c}} \leq t_{\text{f}}$ **then**
13: $t_{\text{C}} = t_{\text{c}}$ and $\mathcal{P}_{\text{MP}} = \hat{\mathcal{P}}_{\text{MP}}$
14: **break**
15: $\lambda_{\text{MP}}(t) = \{1, 2\}, \forall t$
16: **return** $\mathcal{P}_{\text{MP}}$

x_{c} of type P^+P^-, P^-P^+, P^+LP^- or P^-LP^+ from the start point of the trajectory piece to a point on x_{MP} after the discontinuity at t_{D} (line 8 in Algorithm 2). If x_{c} always stays below x_{MP} before t_{c}, the connecting trajectory is accepted and $\mathcal{P}_{\text{MP}}$ and t_{C} are updated (line 13 in Algorithm 2). Algorithm 2 terminates after processing all discontinuities. Assuming that $\mathcal{P}_{\text{MP}}$ initially has m_{MP} trajectory pieces, it follows that there are at most m_{MP} discontinuities in x_{MP}. That is, the number of iterations of the while loop in line 2 is

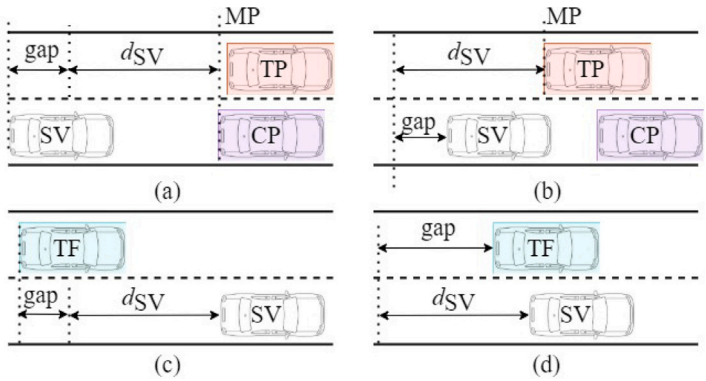

Fig. 12. Possible scenarios before a LC: (a) SV needs to close a gap to MP; (b) SV needs to open a gap to MP; (c) TF is already behind SV; (d) TF needs to open a gap to SV.

$O(m_{MP})$. In addition, the operation in line 3 in Algorithm 2 is $O(m_{MP})$, the operations within the for loop in line 7 in Algorithm 2 are $O(m_{MP})$ and the for loop itself is traversed at most m_{MP} times. Hence, the overall complexity of Algorithm 2 is $O(m_{MP}^3)$.

3.4. Single lane change algorithm

In this section, we develop our LC algorithm for a single SV. To this end, we refer to the vehicles involved in a LC in Fig. 10 and assume that these vehicles start from their initial positions. Hereby, we suppose that the paths $\mathcal{P}_{CP}$ and $\mathcal{P}_{TP}$ of the CP and the TP are already known from a previous computation. We next point out the requirements (desired behavior and constraints) before, during and after a LC. Fig. 12 illustrates the possible relative positions of the SV, MP and TF.

In Fig. 12(a), the SV is at a safe distance behind the MP, whereas the SV is too close to the MP in Fig. 12(b). Similarly, Fig. 12(c) shows the TF at a safe distance behind the SV and the TF is too close to the SV in Fig. 12(d). We next discuss the four possible combinations of relative positions. Hereby, it has to be noted that a LC of the SV at time τ_{SV} can only be initiated if it holds for all $t \in [\tau_{SV}, \tau_{SV} + \Delta_{SV}]$ that

$$x_{SV}(t) + d_{SV} \leq x_{MP}(t) \text{ and } x_{TF}(t) + d_{TF} \leq x_{SV}(t). \tag{16}$$

In combination (a)/(c), the SV already has a safe distance to all surrounding vehicles and can hence immediately start a LC. In addition, it is desired that the SV closes a possible gap to the MP during the LC and approaches the TP after the LC. In the combination (a)/(d), it is required that the TF increases the distance to the SV and in the combination (b)/(c), the SV needs to increase the distance to the MP before a LC can be initiated. Finally, combination (b)/(d) requires both the SV and the TF to increase the distance to their respective predecessors.

We next propose Algorithm 3 to address the stated desired behavior and constraints.

Algorithm 3 first computes the MP trajectory for the given CP and TP (line 1 in Algorithm 3). After that, a path that connects the initial position of the SV to the safe following distance behind the MP is computed with two possible cases. In the first case (line 3 in Algorithm 3), there is extra space between the SV and the MP, which is closed by a forward connection with acceleration a_{max}. In the second case, (line 5 in Algorithm 3), a forward connection with acceleration a_{min} is used to open a sufficient gap behind the MP for the LC of the SV. Similarly, the TF opens a sufficient gap if it is initially too close to the SV (line 7 in Algorithm 3). Hereby, only the connection time t_F is needed in order to determine the LC time of the SV. The actual trajectory of the TF is not computed by Algorithm 3 but as part of the general method in Section 4. Based on the results from the previous computations, the LC time τ_{SV} is determined next. If the SV is initially at a safe distance to the MP, the LC time is determined by the TF (line 11 in Algorithm 3). Otherwise, both t_F and t_{SV} need to be taken into account in order to address (16). If τ_{SV} exceeds the maneuver completion time t_f, a LC is not possible. Then, the SV stays on its initial lane and its trajectory is recomputed according to the trajectory of its predecessor (line 15 in Algorithm 3), and the current leader is updated to the SV (line 16 in Algorithm 3). Otherwise, the computed trajectory x_{SV} is valid until the LC completion time $\tau_{LC} = \tau_{SV} + \Delta_{SV}$. After the LC, the SV should approach and follow the TP at a safe distance. This is realized by computing a forward connection from the SV to the safe following distance behind TP in line 18. After computing the path of the SV, it remains to update the predecessor trajectories on both lanes. As discussed in Section 3.2, the SV disappears from its current lane such that the CP for the following vehicles is given by the SV until the LC completion and by the previous CP otherwise. A suitable trajectory is computed by realizing a forward connection from the SV to the CP at the LC completion time τ_{LC} (line 19 in Algorithm 3). On the target lane, the SV appears behind the TP at the LC start time. That is, a backward connection is used to compute the updated TP path (line 20). Finally, Algorithm 3 assigns the correct leader paths on lane 1 and 2 (line 21 to 24 in Algorithm 3). Since all steps in Algorithm 3 involve algorithms that were shown to terminate with a valid result before (f_{FW} and f_{BW} corresponding to Algorithm 1 and f_{MP} corresponding to Algorithm 2), it is ensured that Algorithm 3 terminates with a valid result. The complexity of Algorithm 3 is dominated by the evaluation of f_{MP} in line 1 and hence $O(m_{MP}^3)$.

Algorithm 3 Algorithm for a single lane change.

$f_{\mathrm{SLC}}(\mathcal{P}_{\mathrm{CP}}, \mathcal{P}_{\mathrm{TP}}, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{TF}}, t_{\mathrm{f}})$
Output: $\mathcal{P}_{\mathrm{SV}}$, τ_{LC}, $\mathcal{P}_{\mathrm{P1}}$, $\mathcal{P}_{\mathrm{P2}}$
1: $\mathcal{P}_{\mathrm{MP}} = f_{\mathrm{MP}}(\mathcal{P}_{\mathrm{CP}}, \mathcal{P}_{\mathrm{TP}})$
2: **if** $x_{\mathrm{SV}}(0) < x_{\mathrm{MP}}(0) - d_{\mathrm{SV}}$ **then**
3: $(\mathcal{P}_{\mathrm{SV}}, t_{\mathrm{SV}}) = f_{\mathrm{FW}}(0, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{MP}} - d_{\mathrm{SV}}, t_{\mathrm{f}}, a_{\max})$
4: **else**
5: $(\mathcal{P}_{\mathrm{SV}}, t_{\mathrm{SV}}) = f_{\mathrm{FW}}(0, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{MP}} - d_{\mathrm{SV}}, t_{\mathrm{f}}, a_{\min})$
6: **if** $x_{\mathrm{TF}}(0) + d_{\mathrm{TF}} > x_{\mathrm{SV}}(0)$ **then**
7: $(\sim, t_{\mathrm{F}}) = f_{\mathrm{FW}}(0, \mathcal{P}_{\mathrm{TF}}, \mathcal{P}_{\mathrm{SV}} - d_{\mathrm{TF}}, t_{\mathrm{f}}, a_{\min})$
8: **else**
9: $t_{\mathrm{F}} = 0$
10: **if** $x_{\mathrm{SV}}(0) + d_{\mathrm{SV}} \leq x_{\mathrm{MP}}(0)$ **then**
11: $\tau_{\mathrm{SV}} = t_{\mathrm{F}}$ and $\tau_{\mathrm{LC}} = \tau_{\mathrm{SV}} + \Delta_{\mathrm{SV}}$
12: **else**
13: $\tau_{\mathrm{SV}} = \max\{t_{\mathrm{SV}}, t_{\mathrm{F}}\}$ and $\tau_{\mathrm{LC}} = \tau_{\mathrm{SV}} + \Delta_{\mathrm{SV}}$
14: **if** $\tau_{\mathrm{SV}} > t_{\mathrm{f}}$ **then**
15: $(\mathcal{P}_{\mathrm{SV}}, \sim) = f_{\mathrm{FW}}(0, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{CP}} - d_{\mathrm{SV}}, t_{\mathrm{f}}, a_{\max})$
16: $\mathcal{P}_{\mathrm{CP}} = \mathcal{P}_{\mathrm{SV}}$, $\tau_{\mathrm{LC}} = \infty$
17: **else**
18: $(\mathcal{P}_{\mathrm{SV}}, t_{\mathrm{SV}}) = f_{\mathrm{FW}}(\tau_{\mathrm{LC}}, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{TP}} - d_{\mathrm{SV}}, t_{\mathrm{f}}, a_{\max})$
19: $(\mathcal{P}_{\mathrm{CP}}, \sim) = f_{\mathrm{FW}}(\tau_{\mathrm{LC}}, \mathcal{P}_{\mathrm{SV}}, \mathcal{P}_{\mathrm{CP}}, t_{\mathrm{f}}, a_{\max})$
20: $(\mathcal{P}_{\mathrm{TP}}, \sim) = f_{\mathrm{BW}}(\tau_{\mathrm{SV}}, \mathcal{P}_{\mathrm{TP}}, \mathcal{P}_{\mathrm{SV}}, 0, a_{\min})$
21: **if** $\lambda_{\mathrm{SV}}(0) = 1$ **then**
22: $\mathcal{P}_{\mathrm{P1}} = \mathcal{P}_{\mathrm{CP}}$, $\mathcal{P}_{\mathrm{P2}} = \mathcal{P}_{\mathrm{TP}}$,
23: **else**
24: $\mathcal{P}_{\mathrm{P1}} = \mathcal{P}_{\mathrm{TP}}$, $\mathcal{P}_{\mathrm{P2}} = \mathcal{P}_{\mathrm{CP}}$,

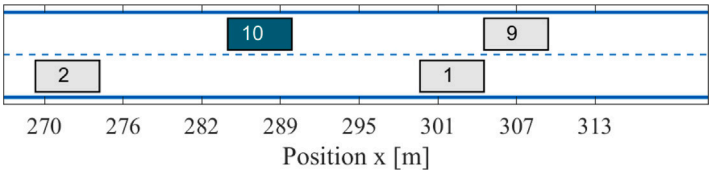

Fig. 13. Single LC example scenario with SV 10, CP 9, TP 1 and TF 2.

An important property of Algorithm 3 is its invariance with respect to the road representation for a generic LC vehicle (denoted as the SV). Specifically, the result of the LC computation is the path of the SV as well as the updated paths of P1 and P2. That is, the same computation can be carried out to determine the path of each subsequent LC vehicle. Our novel algorithm for the computation of multiple simultaneous LCs in Section 4 is based on this important property.

3.5. Illustrative example

We illustrate Algorithm 3 by the example scenario in Fig. 13. Here, the SV with ID 10 performs a LC from lane $\lambda_{10,0} = 2$ to lane $\lambda_{10,\mathrm{f}} = 1$. The CP has ID 9, the TP has ID 1 and the TF has ID 2.

We assume that the paths $\mathcal{P}_{\mathrm{CP}} = \mathcal{P}_9$ and $\mathcal{P}_{\mathrm{TP}} = \mathcal{P}_1$ of the CP and TP are given as shown in Fig. 14 and the path $\mathcal{P}_{\mathrm{MP}}$ is computed in line 1 of Algorithm 3. For illustration, the figure also shows the desired path of the SV (at a distance of $d_{\mathrm{SV}} = 20\,\mathrm{m}$ to the MP) by the yellow dashed line. In addition, the figure shows the slowest possible path of the TF together with the purple dashed line that indicates the desired position of the SV at a distance $d_{\mathrm{TF}} = 20\,\mathrm{m}$ in front of the TF. A LC is only possible if the position of the SV is behind the yellow dashed line and in front of the purple dashed line.

Looking at the initial position of the SV, this corresponds to case (b)/(d) in Fig. 12. That is, a LC is not possible at time $t = 0$ since $x_{\mathrm{SV},0} > x_{\mathrm{MP},0} - d_{\mathrm{SV}}$ and $x_{\mathrm{SV},0} < x_{\mathrm{TF},0} + d_{\mathrm{TF}}$. Hence, the SV has to both slow down in order to reach a safe distance to the MP and wait for the TF to open a sufficient gap for the LC. This task is achieved by computing the forward connection of the SV to the MP in line 5 of Algorithm 3 and by determining the intersection time of x_{SV} and the slowest possible TF path in line 7 of Algorithm 3. Then, the LC start time is computed as $\tau_{\mathrm{SV}} = 4.9\,\mathrm{s}$ in line 13 of Algorithm 3. The final path of the SV is obtained by applying a forward connection to the TP after the LC end time at $\tau_{\mathrm{SV}} + \Delta_{\mathrm{Sv}} = 7.4\,\mathrm{s}$.

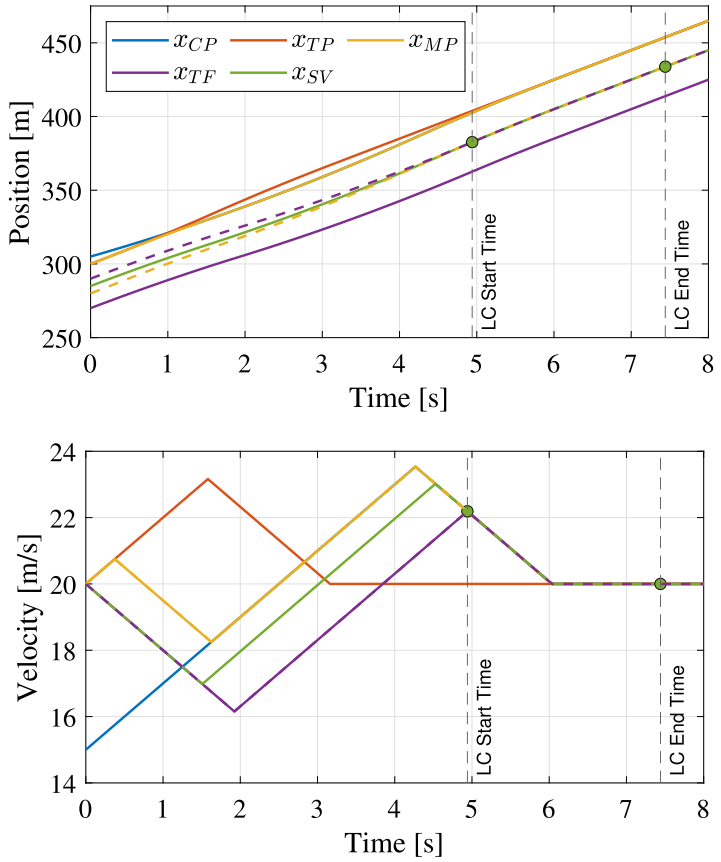

Fig. 14. Single LC example trajectories.

As explained in Section 2.2, we apply the modified version of CACC in order to implement the vehicle paths. That is, the feedforward input of the SV is realized as

$$x_{\mathrm{SV}}^{\mathrm{ff}} = \begin{cases} x_{\mathrm{SV}}(t) - (x_{\mathrm{MP}}(t) - d_{\mathrm{SV}}) & \text{for } 0 \le t \le \tau_{\mathrm{SV}} + \Delta_{\mathrm{SV}} \\ x_{\mathrm{SV}}(t) - (x_{\mathrm{TP}}(t) - d_{\mathrm{SV}}) & \text{otherwise.} \end{cases} \tag{17}$$

Since the SV is already at the desired distance to the TP after the LC, it holds that $x_{\mathrm{SV}}^{\mathrm{ff}}(t) = 0$ for $t \ge \tau_{\mathrm{SV}} + \Delta_{\mathrm{SV}}$ such that the SV applies the classical CACC.

4. Multiple simultaneous lane changes

In this section, we employ Algorithm 3 in order to perform multiple simultaneous LCs. Section 4.1 develops our novel algorithm, which is then illustrated by an example in Section 4.2.

4.1. Multiple LC algorithm

We introduce a list $\mathcal{L}_1$ of vehicles on lane 1, a list $\mathcal{L}_2$ of vehicles on lane 2 and a list $\mathcal{C}$ of LC vehicles. All lists are ordered with a decreasing initial vehicle position. In addition, we initialize the paths $\mathcal{P}_{\mathrm{P1}}$ and $\mathcal{P}_{\mathrm{P2}}$ of the predecessor vehicles P1 and P2 on both lanes using the virtual leader path $\mathcal{P}_0$ with

$$x_{\mathrm{P1}}(t) = x_{\mathrm{P2}}(t) = x_0(t), \tag{18}$$

$$v_{\mathrm{P1}}(t) = v_{\mathrm{P2}}(t) = \dot{x}_0(t) \tag{19}$$

$$\lambda_{\mathrm{P1}}(t) = \{1\}, \lambda_{\mathrm{P2}}(t) = \{2\}. \tag{20}$$

Hereby, x_0 can be any position trajectory as specified in Section 3.1 that fulfills for all $i \in \mathcal{V}$ and $0 \le t \le t_{\mathrm{f}}$ that

$$\hat{x}_{i,0} + \hat{v}_{i,0}\, t + \frac{a_{\min}}{2} t^2 + d_i \le x_0(t) \tag{21}$$

such that $x_0(t)$ can be considered as a safe leader trajectory.

Using this setting, Algorithm 4 presents the details of the proposed LC scheduling method. Hereby, we assume that there is not any ongoing LC when the computations are performed. That is, all LCs are computed by Algorithm 4.

Algorithm 4 Algorithm for multiple lane changes.

$f_{\mathrm{MLC}}(\mathcal{P}_{\mathrm{P1}}, \mathcal{P}_{\mathrm{P2}}, \mathcal{L}_1, \mathcal{L}_2, \mathcal{C}, t_{\mathrm{f}})$
Output: $\mathcal{P}$
1: **Initialize:** $\mathcal{P} = \emptyset$
2: **while true do**
3: **if** $\mathcal{C}$ is empty **then**
4: **for** $k = 1, 2$ **do**
5: **for** vehicles j on L_k starting from front **do**
6: $(\mathcal{P}_j, t_j) = f_{\mathrm{FW}}(0, \mathcal{P}_j, \mathcal{P}_{\mathrm{P}k}, t_{\mathrm{f}}, a_{\max})$
7: Remove j from L_k, $\mathcal{P} = \mathcal{P} \cup \{\mathcal{P}_j\}$, $\mathcal{P}_{\mathrm{P}k} = \mathcal{P}_j$
8: **return** $\mathcal{P}$
9: **else**
10: Remove first LC vehicle i from $\mathcal{C}$
11: $c = \lambda_{i,0}$, $d = \lambda_{i,\mathrm{f}}$
12: **for** vehicle j on L_c with $\hat{x}_{j,0} > \hat{x}_{i,0}$ **do**
13: $(\mathcal{P}_j, t_j) = f_{\mathrm{FW}}(0, \mathcal{P}_j, \mathcal{P}_{\mathrm{P}c}, t_{\mathrm{f}}, a_{\max})$
14: Remove j from L_c, $\mathcal{P} = \mathcal{P} \cup \{\mathcal{P}_j\}$, $\mathcal{P}_{\mathrm{P}c} = \mathcal{P}_j$
15: $\mathcal{I} = \emptyset$, $t_{\min} = \infty$, $\hat{L}_d = L_d$, TF is the first vehicle on $\hat{L}_d$
16: **while true do**
17: $(\mathcal{P}_i, \tau_i, \mathcal{P}_{\mathrm{P1}}, \mathcal{P}_{\mathrm{P2}}) = f_{\mathrm{SLC}}(\mathcal{P}_{\mathrm{P}c}, \mathcal{P}_{\mathrm{P}d}, \mathcal{P}_i, \mathcal{P}_{\mathrm{TF}}, t_{\mathrm{f}})$
18: **if** $\tau_i \le t_{\mathrm{f}} \wedge \tau_i < t_{\min}$ **then**
19: $t_{\min} = \tau_i$, $\hat{\mathcal{I}} = \mathcal{I}$
20: $\hat{\mathcal{P}}_i = \mathcal{P}_i$, $\hat{\mathcal{P}}_{\mathrm{P1}} = \mathcal{P}_{\mathrm{P1}}$, $\hat{\mathcal{P}}_{\mathrm{P2}} = \mathcal{P}_{\mathrm{P2}}$
21: **if** first vehicle on $\hat{L}_d$ is a non-LC vehicle **then**
22: $(\mathcal{P}_{\mathrm{TF}}, t_{\mathrm{TF}}) = f_{\mathrm{FW}}(0, \mathcal{P}_{\mathrm{TF}}, \mathcal{P}_{\mathrm{P}d}, t_{\mathrm{f}})$
23: $\mathcal{I} = \mathcal{I} \cup \{\mathcal{P}_{\mathrm{TF}}\}$, $\mathcal{P}_{\mathrm{P}d} = \mathcal{P}_{\mathrm{TF}}$
24: Remove TF from $\hat{L}_d$
25: TF is the new first vehicle on $\hat{L}_d$
26: **else**
27: **break**
28: **if** $t_{\min} \le t_{\mathrm{f}}$ **then**
29: Remove vehicles in $\hat{\mathcal{I}}$ from $\mathcal{L}_d$ and i from $\mathcal{L}_c$
30: $\mathcal{P} = \mathcal{P} \cup \hat{\mathcal{I}} \cup \{\hat{\mathcal{P}}_i\}$, $\mathcal{P}_{\mathrm{P1}} = \hat{\mathcal{P}}_{\mathrm{P1}}$, $\mathcal{P}_{\mathrm{P2}} = \hat{\mathcal{P}}_{\mathrm{P2}}$
31: **else**
32: $(\mathcal{P}_i, \tau_i) = f_{\mathrm{FW}}(0, \mathcal{P}_i, \mathcal{P}_{\mathrm{P}c}, t_{\mathrm{f}}, a_{\max})$
33: $\mathcal{L}_c = \mathcal{L}_c \setminus \{i\}$, $\mathcal{P} = \mathcal{P} \cup \{\mathcal{P}_i\}$

The main aim of Algorithm 4 is the computation of the group path $\mathcal{P}$ for the given set of vehicles $\mathcal{V}$ that are placed in the lists $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{C}$ as described above. The algorithm starts from an empty group path (line 1 in Algorithm 4) and then first checks if $\mathcal{C}$ is empty (line 3 in Algorithm 4). In the positive case, this means that there are no more LC vehicles. Then, trajectories for all remaining vehicles on both lanes are computed to follow their respective predecessor vehicle (line 4 to 7 in Algorithm 4). If $\mathcal{C}$ is not empty, the algorithm selects the next LC vehicle with index i on $\mathcal{C}$ (line 10 in Algorithm 4). Here, c and d denote the current lane and the target lane of vehicle i, respectively (line 11 in Algorithm 4). Before processing the LC vehicle i, the paths of all vehicles in front of i are computed such that they close potential gaps to their predecessor vehicles (line 12 to 14 in Algorithm 4). The resulting paths of these vehicles are inserted in $\mathcal{P}$ and the vehicles are then removed from $\mathcal{L}_c$ (line 14 in Algorithm 4). The processing of the LC vehicle i requires a set $\mathcal{I}$ that keeps track of the vehicles on the target lane d to be placed in front of i, the minimum LC time $t_{\min}$ and TF vehicle (line 15 in Algorithm 4). Then, a single LC for LC vehicle i is computed (line 17 in Algorithm 4) using Algorithm 3. The parameters of the LC with minimum time for LC vehicle i are updated if the currently computed LC is faster than all previously computed LCs and is completed before t_{f} (line 18 to 20 in Algorithm 4). After that, the next vehicle on $\mathcal{L}_d$ is to be used as the TF. To this end, the current TF performs a forward connection to the predecessor on lane d (line 22 in Algorithm 4), is inserted in $\mathcal{I}$ and its path becomes the predecessor on lane d (line 23 in Algorithm 4). Then, the current TF is removed from $\mathcal{L}_d$ (line 24 in Algorithm 4) and the new first vehicle on $\mathcal{L}_d$ becomes the TF for the next LC computation (line 25 in Algorithm 4). If the first vehicle on $\mathcal{L}_d$ is a LC vehicle, the iteration terminates since our algorithm is designed to avoid LC vehicles overtaking each other (line 27 in Algorithm 4). If the LC computation for LC vehicle i is successful (line 28 in Algorithm 4), the vehicles on lane d and c are updated (line 29 in Algorithm 4). In addition, the computed vehicle paths are inserted in $\mathcal{P}$ and the predecessor vehicles on both lanes are updated (line 30 in Algorithm 4). Hereby, the stored parameters of the LC with the minimum time $t_{\min}$ are used. If the LC is unsuccessful, the path of LC vehicle i is computed such that it follows its predecessor (line 32 in Algorithm 4).

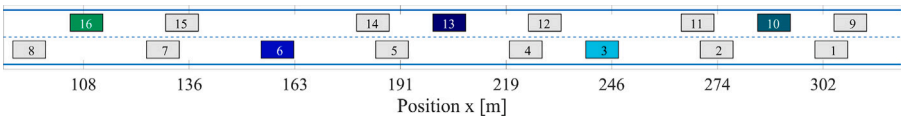

Fig. 15. Multiple LC Example.

Since Algorithm 4 is supposed to be executed in real time, it is important that the algorithm is feasible in the sense of terminating with a valid result for any initial vehicle configuration. Algorithm 4 terminates in line 8. Considering that line 4 to 7 involves simple set operations and algorithms that were shown to terminate previously, termination of Algorithm 4 requires that C becomes empty in line 3 of Algorithm 4. We further note that line 11 to 15 and line 28 to line 33 of Algorithm 4 contain basic operations as well as algorithms that were previously shown to terminate. Consequently, since one LC vehicle is removed from C in line 10 of Algorithm 4 in each iteration of the while loop in line 2 of Algorithm 4, it remains to show that the while loop in line 16 of Algorithm 4 terminates. This while loop removes one vehicle from $\hat{L}_d$ (line 24 in Algorithm 4) as long as the first vehicle on $\hat{L}_d$ is a non-LC vehicle. That is, this while loop terminates in line 27 of Algorithm 4 if $\hat{L}_d$ is empty or the first vehicle on $\hat{L}_d$ is a LC vehicle. Overall, this ensures that Algorithm 4 terminates. Observing that all the computations in Algorithm 4 lead to valid vehicle paths, this ensures that the algorithm is feasible.

Writing n for the number of vehicles and l for the number of LC vehicles, it holds that the number of iterations of the loops in line 2, 12 and 16 in Algorithm 4 is $O(l)$, $O(n)$ and $O(n)$, respectively. In addition, our previous analysis shows that all of the operations carried out within the algorithm are bounded by the complexity $O(m_{\mathrm{MP}}^3)$ of the single LC algorithm. Hence, the overall complexity is $O(n \cdot l \cdot m_{\mathrm{MP}}^3)$.

4.2. Multiple LC illustration

We illustrate Algorithm 4 by the example scenario with the set of vehicles $\mathcal{V} = \{1, \dots, 16\}$ and the list of LC vehicles $C = [10, 3, 13, 6, 16]$ in Fig. 15. We note that the vehicle numbers are assigned from right to left, first on lane 1 and then on lane 2. That is, considering that the direction of travel is from left to right, vehicle 1 is the first vehicle on lane 1 and vehicle 9 is the first vehicle on lane 2. In addition, we assume that the virtual leader travels at a velocity of $v_0 = 20\,\mathrm{m/s}$ starting at a distance of $20\,\mathrm{m}$ ahead of vehicle 9.

In the first iteration of Algorithm 4, the LC vehicle 10 is processed (line 10). First, the position trajectory of vehicle 9 is computed to follow the virtual leader (line 12 to 14 in Algorithm 4) since it is in front of the LC vehicle on the same lane. After that, all possible position trajectories of the LC vehicle on the target lane after the LC are attempted (line 16 to 27 in Algorithm 4): before vehicle 1, between vehicle 1 and 2, between vehicle 2 and 3. Positions behind vehicle 3 are not attempted since vehicle 3 is a LC vehicle. In this case, it is determined that the shortest LC time is obtained when changing the lane between vehicle 1 and 2. That is, the actual LC is executed with the position trajectory of vehicle 9 as the CP and the position trajectory of vehicle 1 as the TP. Although vehicle 2 is used in the computation as the TF, its trajectory is not computed in this iteration. After this iteration, the paths of vehicle 1, 9 and 10 in Fig. 17 are known. Since vehicle 10 is a LC vehicle, it appears on both lanes with an overlap of the shown position trajectories during the LC. In addition, the predecessor vehicle trajectories on both lanes after the LC are determined. Specifically, it holds that the predecessor vehicles on lane 1 and 2 change from the TP (before the LC) to the SV (during and after the LC) and from the SV (until the end of the LC) to the CP (after the LC), respectively. The updated predecessor trajectories are computed using a backward connection on lane 1 and a forward connection on lane 2 as can be seen in Fig. 16. These predecessor trajectories are then used in the second iteration of Algorithm 4, which is concerned with the LC of vehicle 3.

Here, first the position trajectory of vehicle 2 on the same lane as the LC vehicle is computed and then, all positions in front of vehicle 11, between vehicle 11 and 12 and between vehicle 12 and 13 are attempted. The fastest LC is achieved between vehicle 11 and 12 such that the position trajectories of vehicle 2, 3 and 11 in Fig. 17 are known after this iteration in addition to the previously computed paths. In the following iterations, the LC vehicles 13, 6 and 16 are processed successively. Vehicle 13 is placed between vehicle 4 and 5 on lane 1, vehicle 6 is placed between vehicle 14 and 15 on lane 2 and vehicle 15 is placed behind vehicle 7 on lane 1. The final position and velocity trajectories of all vehicles can be seen in Fig. 17, where the latest LC is completed at $\tau_{\mathcal{P}} = 7.3\,\mathrm{s}$.

Remark 2. It is important to note that the iterative computation of the LCs does not imply that the vehicles have to change their lanes successively. This can be seen in the example when looking at the LC vehicle 10. Although vehicle 10 is the first LC vehicle processed, its LC is performed the last. That is, our method adjusts the LC time of each vehicle in order to minimize the time of the latest LC among all vehicles as desired in Section 2.4. This is different from existing methods such as in Li et al. (2018) that require simultaneous LCs of all vehicles.

Remark 3. We also recall that Algorithm 4 has a polynomial complexity in the number of vehicles, LC vehicles and trajectory segments. As will be further illustrated in Section 5, this enables the real-time computation of all vehicle paths.

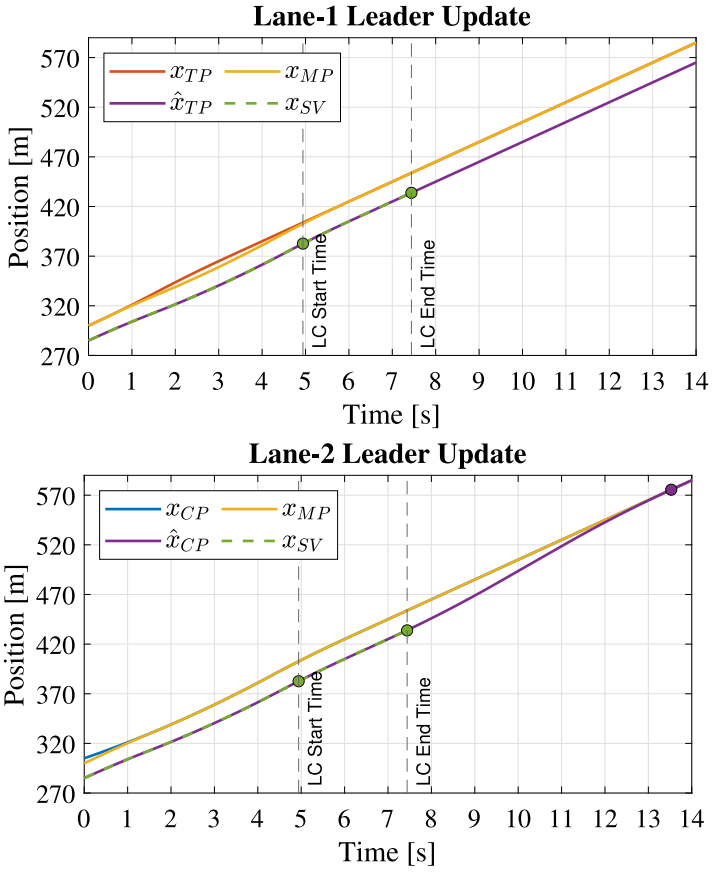

Fig. 16. Updated predecessor trajectories after the LC of vehicle 10.

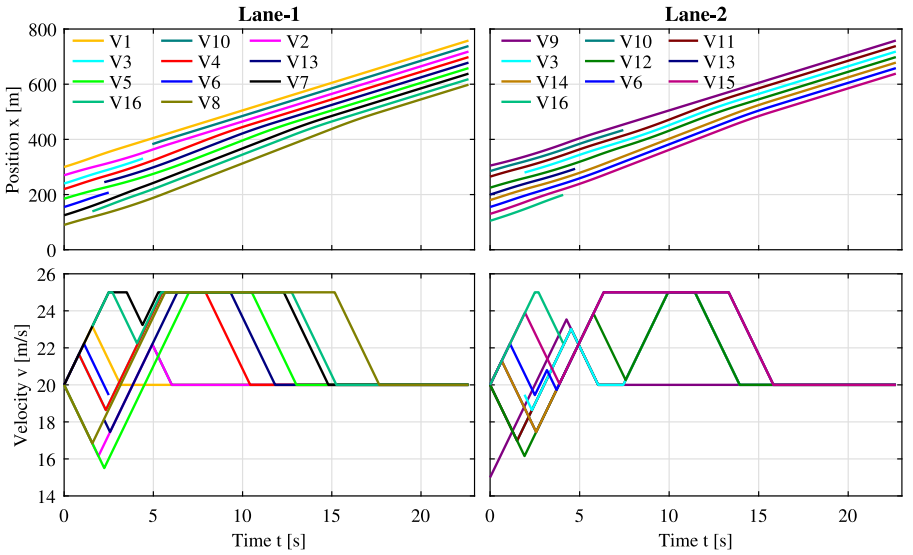

Fig. 17. Reference trajectories for the example with multiple LCs.

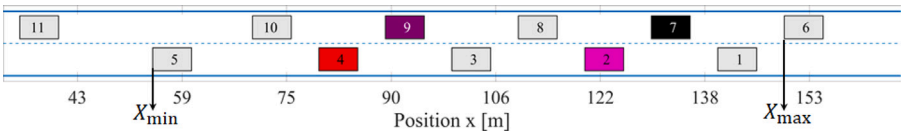

Fig. 18. Minimum speed assignment.

4.3. Adaptation of the minimum velocities

In dense traffic, vehicles follow each other at the desired distance specified in Section 2.2. That is, if one of the vehicles reaches the minimum velocity $v_{\min}$, all of its followers also have to drive at $v_{\min}$. As a result, these vehicles will not be able to open gaps for LCVs until a speed increase, which delays the LCs. In order to address this issue, we propose a modification of our algorithm that defines an individual minimum velocity $v_{i,\min}$ for each vehicle $i \in \mathcal{V}$. To this end, we determine the range of positions that are relevant for LCs, which is given by the maximum vehicle position on any lane

$$X_{\max} = \max_{i \in \mathcal{V}} \{\hat{x}_{i,0}\} \tag{22}$$

and the maximum of the minimum positions on each lane

$$X_{\min} = \max\{\min_{i \in \mathcal{L}_1} \{\hat{x}_{i,0}\}, \min_{j \in \mathcal{L}_j} \{\hat{x}_{j,0}\}\} \tag{23}$$

as illustrated in Fig. 18. Then, we define the minimum velocity coefficient as

$$c_{\min} = \frac{v_{\text{nom}} - v_{\min} - B}{X_{\max} - X_{\min}}. \tag{24}$$

Specifically, $c_{\min}$ indicates how the minimum velocity should be adjusted for each vehicle $i \in \mathcal{V}$ depending on its initial position $\hat{x}_{i,0}$ in the form

$$v_{i,\min} = \begin{cases} v_{\text{nom}} - B - (X_{\max} - x_{\text{i},0}) \cdot c_{\min} & \text{for } \hat{x}_{\text{i},0} \geq X_{\min} \\ v_{\min} & \text{otherwise} \end{cases} \tag{25}$$

Hereby, B is a constant that can be adjusted to achieve a minimum velocity below v_{nom} for the leading vehicles on each lane.

4.4. Discussion

Algorithm 4 computes a group path for a given set of vehicles. Hereby, the resulting vehicle trajectories serve as reference trajectories to be realized using CACC as outlined in Section 2.2. In order to demonstrate the applicability of these reference trajectories, we implemented the CACC method with constant gap policy in Darbha et al. (2020) for the example scenario in Fig. 17 in a simulation experiment. Hereby, the feedforward signals a_i^{ff} and x_i^{ff} for each vehicle i are computed as the position/acceleration difference between the respective predecessor vehicle and vehicle i. The resulting vehicle trajectories are shown in Fig. 19. It is readily observed that the velocity trajectories are smoothened compared to the reference trajectories in Fig. 17 due to the application of CACC. We further emphasize that the proposed method does not require vehicles to directly follow the computed absolute vehicle trajectories, which is susceptible to disturbances. Instead, vehicles apply CACC to achieve safe car following and perform additional maneuvers such as opening/closing gaps relative to their desired path.

It was shown in Section 4.1 that the complexity of Algorithm 4 is $O(n \cdot l \cdot m_{\text{MP}}^3)$, whereby n is the number of vehicles, l is the number of LC vehicles and m_{MP} is the maximum number of trajectory pieces of any MP trajectory. That is, the complexity grows linearly with the number of vehicles. Taking into account that the lane change scheduling setup performs a centralized computation of the vehicle paths, it is required that these vehicle paths are communicated to the vehicles via V2I communication by a roadside unit (RSU). Accordingly, the number of vehicles in a group is bounded by the number of vehicles that can be reached simultaneously by an RSU. Considering that the safe communication range of an RSU is expected to be below 300 m (Reis et al., 2013; Bazzi et al., 2017) and a vehicle can occupy about 25 m on the road (including the inter-vehicle distance), the number of vehicles to be processed by Algorithm 4 is in the order of about 20 vehicles on a two-lane road segment.

We further point out that Algorithm 4 provides vehicle paths for a single group of vehicles. Nevertheless, it is generally the case that there is a continuous flow of vehicles approaching a critical point such as an intersection, a ramp, or a lane closure. In order to address this case, it is possible to apply Algorithm 4 periodically. Each time the algorithm is applied, it first has to be checked which vehicles in the communication range of the RSU are unprocessed. Then, it has to be decided which of the vehicles should form a new group and which vehicles should be merged with the group of vehicles from the previous iteration. In the first case, virtual leaders are introduced and the new vehicles follow those leaders by a direct application of Algorithm 4. In the latter case, the trajectories of the last vehicles on each lane from the previous iteration are used as the leader trajectories for the application of Algorithm 4. Overall, the suggested procedure allows the extension of our method to a continuous stream of vehicles, whereby only a limited number of vehicles that depends on the communication range of the RSU needs to be processed by Algorithm 4.

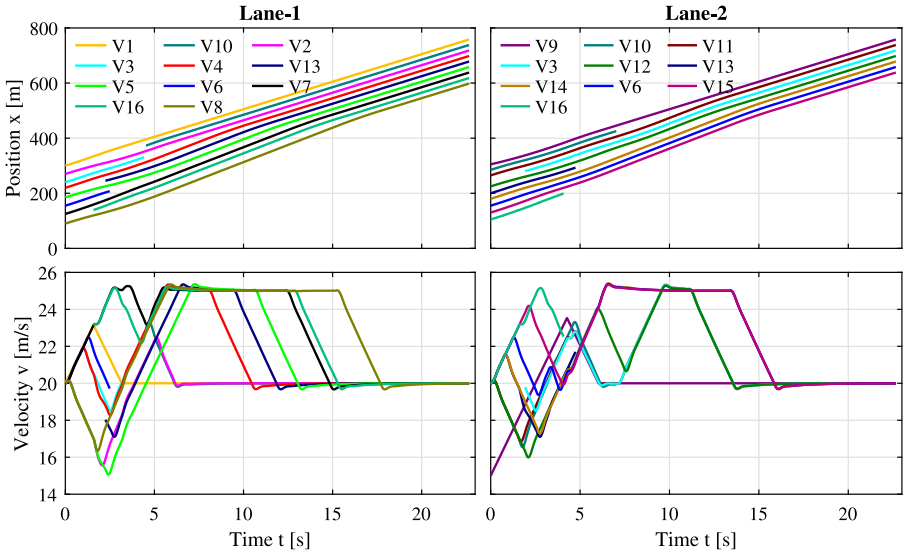

Fig. 19. Vehicle trajectories for the example with multiple LCs when applying CACC.

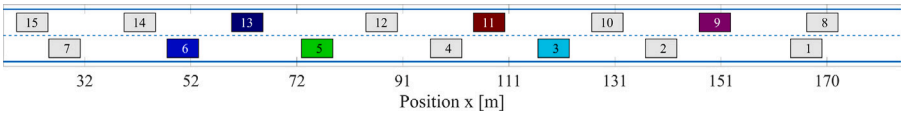

Fig. 20. Example with many LC vehicles and small initial vehicle distances. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Evaluation

This section evaluates the performance of our method. First, Section 5.1 presents several interesting example scenarios. Then, we perform a comparison study with a benchmark method in the literature based on computational experiments. Section 5.2 briefly outlines the benchmark method and introduces our performance metrics. The results of our comparison study are presented in Section 5.3.

5.1. Example cases

In the first example, we assume that 15 vehicles in the set $\mathcal{V} = \{1, \dots, 15\}$ are positioned in a compact manner as shown in Fig. 20, whereby the LC vehicles $C = [9, 3, 11, 5, 13, 6]$ are shown in color.

We perform two experiments with this configuration. The result for the application of Algorithm 4 with the same minimal velocity $v_{\min}$ for all vehicles is shown in Fig. 21, whereas the corresponding result when computing the minimal velocity as in Section 4.3 with $B = 1\,\text{m/s}$ can be seen in Fig. 22. In both settings the virtual leader is located at a distance of 20 m ahead of vehicle 8, and travels at a velocity of 20 m/s.

It is readily observed that the LC completion time is $\tau_{\mathcal{P}} = 20.07\,\text{s}$ in Fig. 21 and $\tau_{\mathcal{P}} = 16.18\,\text{s}$ in Fig. 22. That is, using different minimum velocities indeed has a positive effect on the LC completion time since all CAVs are able to open gaps as is confirmed when comparing the velocity plots of Figs. 21 and 22.

The previous experiment considered a leader vehicle driving with the constant velocity v_{nom}. In the next experiment, we show that our method is also able to schedule LCs in the case where the virtual leader is placed very close, only 3 m's ahead of vehicle 1, and its velocity changes over time, starting with an initial velocity of 20 m/s, then slowing down to 19 m/s, subsequently accelerating to 20 m/s and then to 25 m/s, and then returning to 20 m/s. Close positioning of the virtual leader makes vehicle 1 decelerate in order to adjust its relative position to the virtual leader as can be seen in Fig. 23. It is readily observed that our algorithm successfully schedules the LCs of all CAVs until $\tau_{\mathcal{P}} = 20.12\,\text{s}$.

Finally, we demonstrate that our method can as well handle conflict zones such as lane closures. As is depicted in Fig. 24, all (colored) vehicles on lane 1 are supposed to perform a LC. The virtual leader positioning is the same as in the example in Fig. 21 The result when applying Algorithm 4 to this case is also shown in the figure. It can be seen that all vehicles are able to change their lane until time $\tau_{\mathcal{P}} = 31.39\,\text{s}$, whereby the vehicles on lane 2 open the required gaps for the vehicles on lane 1 for safe merging.

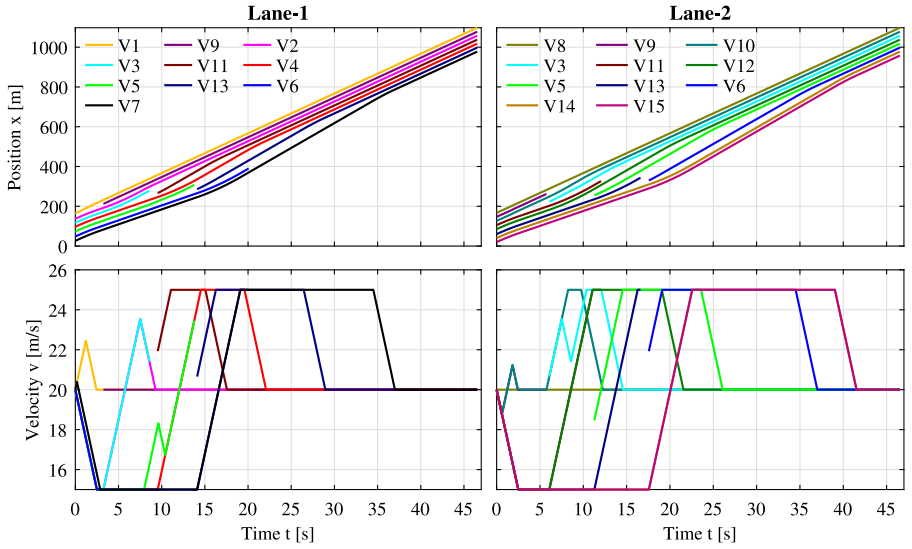

Fig. 21. Vehicle trajectories for the scenario in Fig. 20: all vehicles have the same $v_{\min}$.

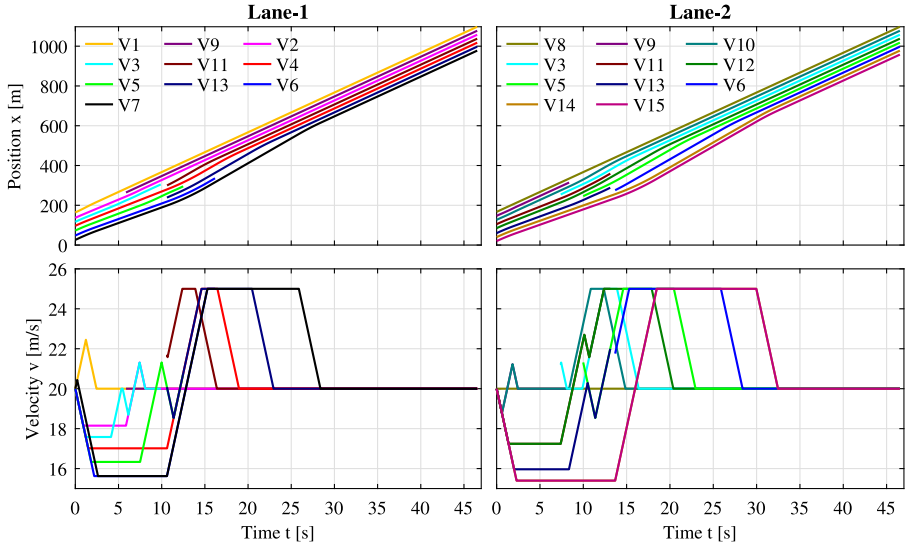

Fig. 22. Vehicle trajectories for the scenario in Fig. 20: $v_{\min}$ is computed using the method in Section 4.3.

5.2. Benchmark method and performance metrics

In order to assess the performance of our method, we compare our results to the method by Li et al. (2018), which addresses the LC scheduling problem in two stages. In the first stage, a sparse configuration is obtained by opening gaps between CAVs. This is achieved by first computing LC trajectories for all LC-CAVs while ignoring potential collisions. Next, the obtained trajectories are translated such that they do not overlap and all the collisions are recovered. This gives *a sparse formation*. Then, an optimization problem is solved to realize the transition from the actual to the sparse formation. In the second stage, all the LC's are performed simultaneously in a safe manner, since sufficient gaps are opened.

For illustration, Fig. 25 shows the result of the application of the benchmark method to the scenario in Fig. 20. Here, it can be seen that all the LCs are performed and completed simultaneously at time 18.43 s. In addition, it has to be noted that the position 304.6 m of the last CAV at the end of the LCs is small since all LCs are carried out simultaneously in a sparse formation. In comparison, our method leads to a much more compact configuration with a position of 331.7 m of the last CAV, since CAVs can already close gaps while other CAVs still perform LCs.

In order to achieve a detailed comparison, we next perform different experiments. In these experiments, CAVs on two lanes are generated randomly with different ranges of initial vehicle distances: 15–17 m, 15–20 m, 15–30 m, 15–45 m and 15–60 m, whereby

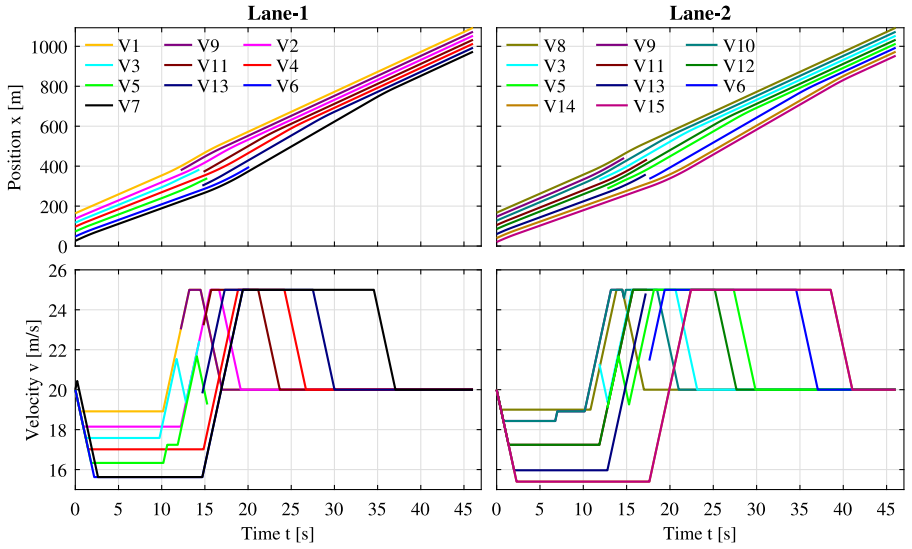

Fig. 23. Example case with variable velocity of the leader vehicle.

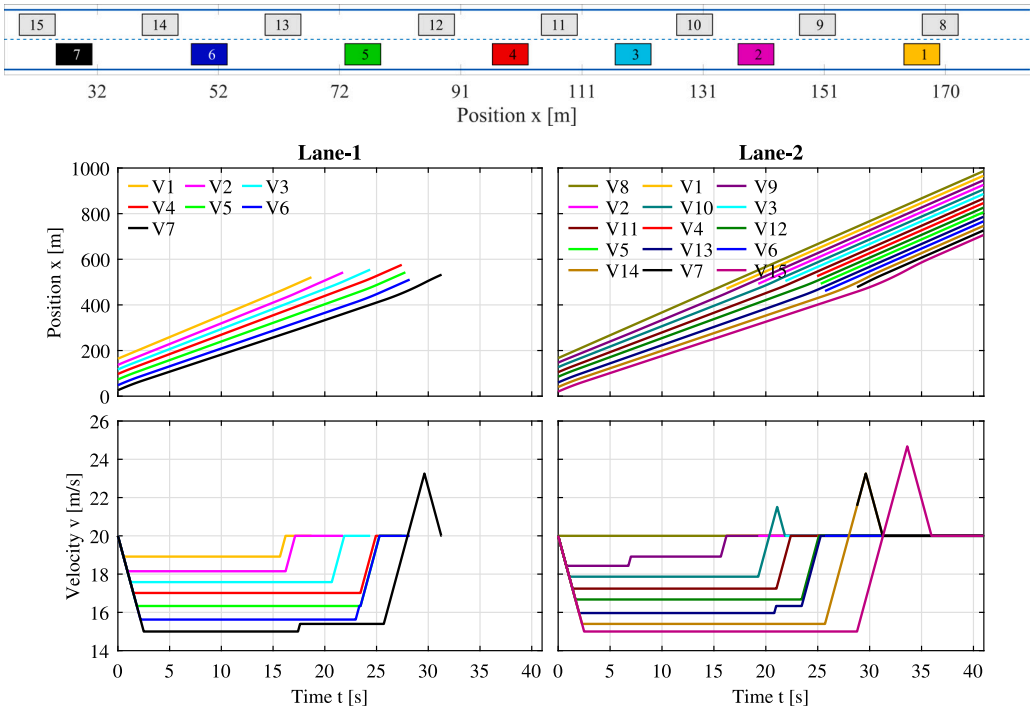

Fig. 24. Lane closure example. All vehicles move to lane 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

15 m represents the minimum safety distance. For each case, 200 test scenarios with 20 CAVs each are created, the speed values $v_{\min} = 15$ m/s, $v_{\text{nom}} = 20$ m/s and $v_{\max} = 25$ m/s, and the acceleration values $a_{\min} = -2$ m/s^2, $a_{\max} = 2$ m/s^2 are used. We select three performance metrics for the comparison. First, we consider the LC completion time $\tau_{\mathcal{P}}$ of the group path $\mathcal{P}$, which is the first objective of our optimization problem in (12). Second, we determine the position x_{last} of the last CAV at the lane change completion time. This metric characterizes the overall progress of the group of CAVs in the driving direction and relates to the second objective of our optimization problem in (12). Third, we capture the algorithmic runtime of the LC computation, which is important to assess the real-time capabilities of our method. All experiments were carried out using Matlab 2020b on a laptop with Intel(R) Core(TM) i7-4510U CPU and 16 GB RAM. The computations for the benchmark method were performed based on the code provided by Li

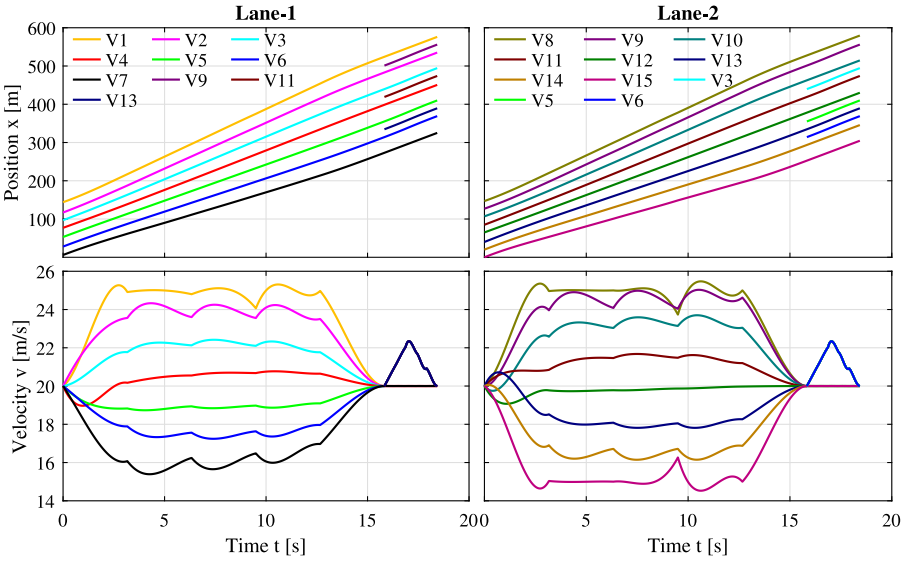

Fig. 25. Computed trajectories of the benchmark method for the example case in Fig. 20.

Table 1
Comparison with the benchmark method: τ_p and x_{last}.

Case	Constant $v_{\min}$		Variable $v_{\min}$	
	#Wins	Improvement	#Wins	Improvement
LC completion time τ_p				
15–17 m	149 (74%)	1.00 s	193 (96%)	2.25 s
15–20 m	145 (72%)	0.94 s	192 (95%)	2.35 s
15–30 m	141 (70%)	1.15 s	181 (90%)	2.26 s
15–45 m	160 (80%)	0.95 s	149 (74%)	0.83 s
15–60 m	152 (76%)	0.55 s	141 (70%)	0.4 s
Last CAV position x_{last}				
15–17 m	194 (97%)	45.40 m	196 (98%)	48.40 m
15–20 m	194 (97%)	41.14 m	196 (98%)	45.14 m
15–30 m	198 (99%)	31.64 m	200 (100%)	33.34 m
15–45 m	199 (99.5%)	21.38 m	198 (99%)	22.02 m
15–60 m	198 (99%)	19.09 m	198 (99%)	19.88 m

(2018), with the speed values $v_{\text{st}} = 20\,\text{m/s}$ and $v_{\max} = 25\,\text{m/s}$, the acceleration values $a_{\min} = -2\ \text{m/s}^2$, $a_{\max} = 2\,\text{m/s}^2$, and the jerk values $j = \pm 1000\ \text{m/s}^3$.

5.3. Results

The comparison results for the LC completion time τ_p and the position x_{last} of the last CAV at τ_p are shown in Table 1.

Here, "#Wins" indicates the number of times our method outperforms the benchmark method by Li et al. (2018) and "Improvement" is the average improvement of the respective quantity. Regarding τ_p, it can be observed that Algorithm 4 performs better than the benchmark method in more than 70% of the cases, even if the same $v_{\min}$ is chosen for all CAVs. Hereby, τ_p is reduced in the order of 1 s. This improvement increases to more than 2 s if different values of $v_{\min}$ are used for different CAVs as described in Section 4.3. Hereby, it has to be noted that larger improvements are obtained for dense initial configurations. This is expected since gaps for LC-CAVs have to be opened, whereas gaps are already available in less dense configurations. When looking at x_{last}, it is apparent that our method outperforms the method by Li et al. (2018) in almost all cases with improvements in the order of 20 m to almost 50 m. The reason for the superiority of our method is the possibility of CAVs closing gaps at all times, whereas the benchmark method requires a sparse formation, where no gaps have been closed.

We finally look at the average run-times of our method and the benchmark method in Table 2. It can be seen that the runtimes of our method are in the order of 200 ms in all cases, whereas the runtimes of the benchmark method are mostly above 10 s. In particular, it turns out that our method is orders of magnitudes faster than the benchmark method in all test cases. This is due to the fact that our method does not require nonlinear optimization and is hence suitable for a real-time implementation.

Table 2
Average algorithmic run-times.

Case	Constant $v_{\min}$	Variable $v_{\min}$	Benchmark
15–17 m	0.16 s	0.20 s	9.97 s
15–20 m	0.18 s	0.20 s	12.65 s
15–30 m	0.15 s	0.17 s	15.81 s
15–45 m	0.18 s	0.15 s	20.32 s
15–60 m	0.19 s	0.14 s	19.00 s

6. Conclusion

In this paper, we consider the scheduling of lane changes of a group of connected and autonomous vehicles (CAVs) that are approaching a critical point such as an intersection, a ramp, or a lane closure. Hereby, the aim is to minimize the time when all lane changes are completed, while keeping the inter-vehicle distances as small as possible in order to ensure both driving safety and high traffic throughput. The main building block of our method is a novel algorithm for performing a safe lane change of a single CAV in the shortest possible time. This algorithm is then applied repeatedly to all lane-changing CAVs to obtain trajectories for all vehicles in the considered group of CAVs. In this context, it is important to note that our algorithm computes CAV trajectories analytically based on a second-order vehicle model such that all computations can be carried out in real time. Moreover, driving safety and tight car following are achieved by implementing the computed CAV trajectories using cooperative adaptive cruise control. In order to evaluate our method, we perform detailed simulation experiments. These experiments show that our method efficiently computes CAV trajectories even in difficult cases with a large number of lane-changing vehicles and arbitrary maneuvers of the considered group of CAVs. In addition, a comparison with a benchmark method highlights the benefits of our method regarding real-time computations and maintaining tight vehicle formations.

The results of this paper suggest several directions for future research. First, we are currently working on the application of the proposed method for interleaving groups of vehicles at traffic intersections. Second, we are planning to use our method for high-level traffic planning on highways as well as arterial roads.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

An, H., il Jung, J., 2018. Design of a cooperative lane change protocol for a connected and automated vehicle based on an estimation of the communication delay. Sensors 18 (10), 3499.

Atagoziyev, M., Schmidt, K.W., Schmidt, E.G., 2016. Lane change scheduling for autonomous vehicles*. IFAC-PapersOnLine 49 (3), 61–66. http://dx.doi.org/10.1016/j.ifacol.2016.07.011, 14th IFAC Symposium on Control in Transportation Systems, CTS 2016, Istanbul, Turkey, 18 – 20 May 2016 URL http://www.sciencedirect.com/science/article/pii/S2405896316302063.

Barria, J.A., Thajchayapong, S., 2011. Detection and classification of traffic anomalies using microscopic traffic variables. IEEE Trans. Intell. Transp. Syst. 12 (3), 695–704. http://dx.doi.org/10.1109/TITS.2011.2157689.

Bazzi, A., Masini, B.M., Zanella, A., Thibault, I., 2017. On the performance of IEEE 802.11 p and LTE-V2V for the cooperative awareness of connected vehicles. IEEE Trans. Veh. Technol. 66 (11), 10419–10432.

Cao, D., Wu, J., Wu, J., Kulcsár, B., Qu, X., 2021. A platoon regulation algorithm to improve the traffic performance of highway work zones. Comput.-Aided Civ. Infrastruct. Eng. 36 (7), 941–956.

Chouhan, A.P., Banda, G., Jothibasu, K., 2020. A cooperative algorithm for lane sorting of autonomous vehicles. IEEE Access 8, 88759–88768.

Darbha, S., Konduri, S., Pagilla, P.R., 2020. Vehicle platooning with constant spacing strategies and multiple vehicle look ahead information. IET Intell. Transp. Syst. 14 (6), 589–600.

di Bernardo, M., Falcone, P., Salvi, A., Santini, S., 2015. Design, analysis, and experimental validation of a distributed protocol for platooning in the presence of time-varying heterogeneous delays. IEEE Trans. Control Syst. Technol. 24 (2), 413–427.

Divakarla, K.P., Emadi, A., Razavi, S., Habibi, S., Yan, F., 2019. A review of autonomous vehicle technology landscape. Int. J. Electr. Hybrid Veh. 11 (4), 320–345.

Eskandarian, A., Wu, C., Sun, C., 2019. Research advances and challenges of autonomous and connected ground vehicles. IEEE Trans. Intell. Transp. Syst..

Hoberock, L.L., 1977. A Survey of Longitudinal Acceleration Comfort Studies in Ground Transportation Vehicles. J. Dyn. Syst. Meas. Control 99 (2), 76–84. http://dx.doi.org/10.1115/1.3427093, arXiv:https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/99/2/76/5778098/76_1.pdf.

Hu, X., Sun, J., 2019. Trajectory optimization of connected and autonomous vehicles at a multilane freeway merging area. Transp. Res. C 101, 111–125. http://dx.doi.org/10.1016/j.trc.2019.02.016, URL http://www.sciencedirect.com/science/article/pii/S0968090X18304844.

Johnson, J., Hauser, K., 2012. Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In: 2012 IEEE International Conference on Robotics and Automation. IEEE, pp. 2035–2041.

Li, B., 2018. Cooperative-lane-changes-of-CAVs. URL https://github.com/libai1943/Cooperative-Lane-Changes-of-CAVs.

Li, T., Wu, J., Chan, C.-Y., Liu, M., Zhu, C., Lu, W., Hu, K., 2020. A cooperative lane change model for connected and automated vehicles. IEEE Access 8, 54940–54951. http://dx.doi.org/10.1109/ACCESS.2020.2981169.

Li, B., Zhang, Y., Feng, Y., Zhang, Y., Ge, Y., Shao, Z., 2018. Balancing computation speed and quality: A decentralized motion planning method for cooperative lane changes of connected and automated vehicles. IEEE Trans. Intell. Veh. 3 (3), 340–350.

Lin, D., Li, L., Jabari, S.E., 2019. Pay to change lanes: A cooperative lane-changing strategy for connected/automated driving. Transp. Res. C 105, 550–564. http://dx.doi.org/10.1016/j.trc.2019.06.006, URL http://www.sciencedirect.com/science/article/pii/S0968090X18310945.

Lu, G., Nie, Y.M., Liu, X., Li, D., 2019. Trajectory-based traffic management inside an autonomous vehicle zone. Transp. Res. B 120, 76–98.
Luo, Y., Xiang, Y., Cao, K., Li, K., 2016. A dynamic automated lane change maneuver based on vehicle-to-vehicle communication. Transp. Res. C 62, 87–102. http://dx.doi.org/10.1016/j.trc.2015.11.011, URL http://www.sciencedirect.com/science/article/pii/S0968090X15004118.
Malik, S., Khan, M.A., El-Sayed, H., 2021. Collaborative autonomous driving—A survey of solution approaches and future challenges. Sensors 21 (11), http://dx.doi.org/10.3390/s21113783, URL https://www.mdpi.com/1424-8220/21/11/3783.
Mariani, S., Cabri, G., Zambonelli, F., 2021. Coordination of autonomous vehicles: Taxonomy and survey. ACM Comput. Surv. 54 (1), 1–33.
Martínez-Díaz, M., Soriguera, F., 2018. Autonomous vehicles: Theoretical and practical challenges. Transp. Res. Procedia 33, 275–282.
Matsuzaki, R., Kamai, K., Seki, R., 2014. Intelligent tires for identifying coefficient of friction of tire/road contact surfaces using three-axis accelerometer. Smart Mater. Struct. 24 (2), 025010.
Nandi, A.K., Chakraborty, D., Vaz, W., 2015. Design of a comfortable optimal driving strategy for electric vehicles using multi-objective optimization. J. Power Sources 283, 1–18. http://dx.doi.org/10.1016/j.jpowsour.2015.02.109, URL https://www.sciencedirect.com/science/article/pii/S0378775315003547.
Nguyen, T., Au, T., 2019. A constant-time algorithm for checking reachability of arrival times and arrival velocities of autonomous vehicles. In: 2019 IEEE Intelligent Vehicles Symposium. IV, pp. 2039–2044. http://dx.doi.org/10.1109/IVS.2019.8813844.
Nie, J., Zhang, J., Ding, W., Wan, X., Chen, X., Ran, B., 2017. Decentralized cooperative lane-changing decision-making for connected autonomous vehicles*. IEEE Access 4, 9413–9420. http://dx.doi.org/10.1109/ACCESS.2017.2649567.
Reis, A.B., Sargento, S., Neves, F., Tonguz, O.K., 2013. Deploying roadside units in sparse vehicular networks: What really works and what does not. IEEE Trans. Veh. Technol. 63 (6), 2794–2806.
Sağlam, H.B., Schmidt, K.W., 2018. Outputs bounds for linear systems with repeated input signals: Existence, computation and application to vehicle platooning. Turk. J. Electr. Eng. Comput. Sci. 26 (1), 283–293.
Shawky, M., 2020. Factors affecting lane change crashes. IATSS Res. 44 (2), 155–161. http://dx.doi.org/10.1016/j.iatssr.2019.12.002, URL https://www.sciencedirect.com/science/article/pii/S0386111219300020.
Wang, Z., Zhao, X., Chen, Z., Li, X., 2021. A dynamic cooperative lane-changing model for connected and autonomous vehicles with possible accelerations of a preceding vehicle. Expert Syst. Appl. 173, 114675.
Wu, Z., Liu, Y., Pan, G., 2009. A smart car control model for brake comfort based on car following. IEEE Trans. Intell. Transp. Syst. 10 (1), 42–46.
Xu, H., Zhang, Y., Cassandras, C.G., Li, L., Feng, S., 2020. A bi-level cooperative driving strategy allowing lane changes. Transp. Res. C 120, 102773.
Zhang, L., Orosz, G., 2016. Motif-based design for connected vehicle systems in presence of heterogeneous connectivity structures and time delays. IEEE Trans. Intell. Transp. Syst. 17 (6), 1638–1651.
Zheng, Z., 2014. Recent developments and research needs in modeling lane changing. Transp. Res. B 60, 16–32. http://dx.doi.org/10.1016/j.trb.2013.11.009, URL https://www.sciencedirect.com/science/article/pii/S019126151300218X.
Zheng, Z., Ahn, S., Chen, D., Laval, J., 2011. Freeway traffic oscillations: Microscopic analysis of formations and propagations using wavelet transform. Transp. Res. B 45 (9), 1378–1388. http://dx.doi.org/10.1016/j.trb.2011.05.012, Select Papers from the 19th ISTTT URL https://www.sciencedirect.com/science/article/pii/S0191261511000622.
Zheng, Y., Ran, B., Qu, X., Zhang, J., Lin, Y., 2020. Cooperative lane changing strategies to improve traffic operation and safety nearby freeway off-ramps in a connected and automated vehicles environment. IEEE Trans. Intell. Transp. Syst. 21 (11), 4605–4614. http://dx.doi.org/10.1109/TITS.2019.2942050.

Maksat Atagoziev received his B.S. degree in computer engineering from Kyrgyz State Technical University, Bishkek, Kyrgyzstan, in 2002 and his M.S. and Ph.D. degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2008 and 2022, respectively. His research interests include intelligent transportation systems, vehicle maneuver coordination and scheduling.

Ece Güran Schmidt received her B.S. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 1997 and her M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2001 and 2004, respectively. She is currently a Faculty Member with the Department of Electrical and Electronics Engineering, Middle East Technical University. Her research interests include high-speed networks, networked control systems, and vehicular communication networks.

Klaus Werner Schmidt received the Diploma and Ph.D. degrees from the University of Erlangen-Nürnberg, Germany, in 2002 and 2005, respectively, both in Electrical, Electronic, and Communication Engineering. He is currently a Professor at the Department of Electrical & Electronics Engineering, Middle East Technical University, Ankara.

His research interests include supervisory control for discrete event systems, industrial automation systems, industrial communication networks, intelligent transportation systems and industrial project control. He is an Associate Editor for Discrete Event Dynamic Systems.