Transportation Research Part C 145 (2022) 103952

Contents lists available at ScienceDirect

Transportation Research Part C

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

Strategic and tactical decision-making for cooperative vehicle platooning with organized behavior on multi-lane highways

Xu Han [a], Runsheng Xu [a], Xin Xia [a], Anoop Sathyan [b], Yi Guo [b], Pavle Bujanović [c], Ed Leslie [d], Mohammad Goli [e], Jiaqi Ma [a,*]

[a] Department of Civil and Environmental Engineering, University of California, Los Angeles, CA 90095, USA
[b] Department of Aerospace Engineering, University of Cincinnati, Cincinnati, OH 45221, USA
[c] US Department of Transportation, Federal Highway Administration, Mclean VA, 22101, USA
[d] Leidos, Mclean, VA 22101, USA
[e] Noblis, Inc., Reston, VA 22101, USA

ARTICLE INFO

Keywords:
Connected and automated vehicle (CAV)
Cooperative driving automation
Cooperative driving
Automated driving system (ADS)
Platooning
Finite state machine (FSM)
Genetic fuzzy system (GFS)

ABSTRACT

Driving automation and vehicle-to-vehicle (V2V) communication provide opportunities to deploy cooperative automated driving systems (C-ADS) for transportation system goals such as sustainability, safety, and efficiency. Among various C-ADS applications, vehicle platooning has great potential to achieve the above system management goals by establishing trajectory-aware V2V cooperative strategies among C-ADS vehicles. Previously, the concept of cooperative adaptive cruise control (CACC)—that is, single-lane decentralized ad-hoc operations of multiple vehicles that closely follow each other—has been studied by researchers extensively. This study builds upon the existing research and proposes a comprehensive multi-lane platooning algorithm with organized behavior via a hierarchical framework. The proposed algorithm adopts the modern state of the art (SOTA) C-ADS software platform framework, which consist of perception, plan and control levels. The multi-lane platooning algorithm incorporates both the strategic level (i.e., mission level) and the tactical level (i.e., motion level) decision-making to cope with complex multi-lane highway challenges, including same-lane platooning, multi-lane joining, and on-ramp merging. Based on the algorithm's strategies, the platoon leaders coordinate between platoon members and external vehicles to guide the platoon through complicated and realistic driving scenarios. On the strategic mission level, a platooning behavior protocol based on a deterministic finite state machine (FSM) is developed to guide the member operations. Additionally, as heuristic protocols fall short in explicitly expressing complex cooperative scenarios, a genetic fuzzy system was trained with FSM as a baseline to extend the algorithm's capability under the cooperative on-ramp merge scenarios. On the tactical motion level, trajectory generation for general ADS maneuvers (i.e., lane following and lane changing) and platooning behavior regulation is proposed such that planned trajectories of other relevant vehicles can be fully considered (i.e., intent sharing of predictive nature). The performance is evaluated in both traffic and automated driving simulators, and the results indicate that the proposed comprehensive multi-lane platooning algorithm can efficiently and safely regulate C-ADS-equipped vehicle behavior and meet system goals.

Peer review under responsibility of Please add a footnote in the below mentioned format for all special issue papers with item group description as "VSI: <special issue title>"This article belongs to the Virtual Special Issue on "special issue full title."

* Corresponding author.
E-mail address: jiaqi.ma@ucla.edu (J. Ma).

https://doi.org/10.1016/j.trc.2022.103952
Received 23 May 2022; Received in revised form 28 September 2022; Accepted 31 October 2022
Available online 12 November 2022

1. Introduction

Intelligent Transportation Systems (ITS) are reshaping transportation and have demonstrated a promising potential to elevate transportation system management, operations, safety, and efficiency. As one of the essential sub-fields in ITS, Cooperative Driving Automation (CDA) is defined in SAE J3216 (SAE International, 2020) and refers to vehicle automation that uses machine-to-machine communication to enable cooperation among two or more entities (e.g., vehicles, infrastructure components). Traffic efficiency, energy consumption, and driving safety can be significantly improved (Stevens and Hopkin, 2012) with the potential benefit of significantly enhanced perception performance (i.e., perception accuracy and perception range) (Xu et al., 2021).

In previous studies, the concept of cooperative adaptive cruise control (CACC), i.e., single-lane decentralized ad-hoc operations of multiple vehicles that closely follow each other, has been extensively studied (Guo and Ma, 2020; Milanés and Shladover, 2014; Adebisi et al., 2020) to enhance the longitudinal behavior of advanced driver-assistance system via cooperation. Several highway traffic simulations conducted by the California Partners for Advanced Transportation Technology (Guo and Ma, 2020; Milanés and Shladover, 2014) showed that autonomous adaptive cruise control (ACC) alone had little effect on lane capacity, even at high market penetration rates (MPRs). On-the-road experiments (e.g., Milanés and Shladover, 2014) showed that a stream of autonomous ACC vehicles is string unstable and more likely to restrain the lane capacity. Therefore, the combination of vehicle-to-vehicle (V2V) communication and ACC system allows an ADS-equipped vehicle to leverage the information from the preceding automated driving systems (ADS)-equipped vehicle to maintain an optimal following distance, leading to a string stable vehicle string. The primary improvement brought by CACC is reducing traffic congestion by improving highway capacity, increasing throughput, and attenuating traffic flow disturbances. In particular, the deployment of V2V communication has the potential to reduce the mean following time gap from about 1.4 s when driving manually to approximately 0.6 s when using CACC (Milanés and Shladover, 2014), significantly increasing highway lane capacity. Notably, because CACC systems allow much narrower following distance by establishing a V2V communication hierarchy, the lane capacity could be increased from 2,200 vehicles per hour (vph) to almost 4,000 vph at 100 percent market penetration (Guo and Ma, 2020; Milanés and Shladover, 2014).

One major drawback of the existing CACC vehicle string is the lack of explicit consideration of lateral V2V cooperation. The CACC's capability of handling cut-in HDVs has been enhanced in a later study (Milanés and Shladover, 2015), but the improvement is limited to handling cut-in human-driven vehicles (HDVs) where the controller only considers the longitudinal direction. Various studies have recently focused on CACC applications in multi-lane settings (i.e., considered lateral control). Firoozi (Firoozi et al., 2021) utilized a pre-defined lookup table to optimize platooning behaviors among multiple lanes. The platoon can switch formation between single or multiple adjacent lanes to avoid congestion under the leader's command. This work established a leader–follower relationship but the lack of intent-sharing limit this algorithm as a passive reformation method with a fixed number of members. Pizarro et al. (2021) proposes a distributed scheme for lane changes inside a platoon that harnesses graph theoretical formulation for reference generation (during multi-lane formation) and distributed model predictive control (DMPC) for vehicle commands to achieve smooth multi-lane platoon maneuvers. The intentions of the peer members are shared and considered because each connected and automated vehicle (CAV) tracks relative state differences with regard to its neighbors. However, both multi-lane platooning methods assume a fixed platoon length, only managing existing vehicles within the platoon. Meanwhile, none of the existing work considers complex cooperation scenarios such as on-ramp mergers. In this regard, Uno's team (Uno, A., 1999) was the first to apply the "virtual vehicle" concept whereby a "ghost" leading CAV was mapped onto the highway before the merging maneuver happened, allowing a safer and smoother lane change control. Later, a distributed consensus-based highway on-ramp merging system (Wang et al., 2018) was developed, where two connected CACC systems are formed on the mainline and on-ramp, further enhancing the on-ramp merging bottleneck. In particular, the merging sequence is determined based on the estimated time-to-arrival calculated by road side unit, whereby each vehicle's intention (i.e., merging time and speed) is estimated. However, the work lacks active member management, which, when combined with intend-sharing, supports intention-based V2V cooperation such as joining and departing from a platoon. In addition, most current on-ramp merging studies either follow simple sequencing logic (i.e., first-in-first-out) or directly assume the existence of a near optimal sequence, leaving another research gap for an integrated multi-lane platooning algorithm that can assign near-optimal merging sequence with the objective to improve network capacity.

On the other hand, extensive studies (Soleimaniamiri et al., 2021; Kato et al., 2018; Fan et al., 2018) have been conducted on developing and deploying comprehensive state-of-the-art (SOTA) ADS platforms (i.e., end-to-end software stacks including perception, planning, and control modules) on capable vehicles to accomplish ADS functions and V2V communication. The SOTA ADS platforms have unique frameworks, in which the planning module includes a mission planning step configuring the semantic goal and a motion planning step that generates the corresponding trajectories. Such a framework allows the planning module to interact and cooperate with other cooperative automated driving systems (C-ADS) modules, such as the perception, control modules, and drive-by-wire modules, to achieve full ADS functions on a real-world C-ADS-equipped vehicle.

The necessity of developing and utilizing the SOTA ADS platform is prominent. For one, microscopic simulators, such as SUMO (Lopez et al., 2018) and VISSIM (Fellendorf et al., 2010), directly assign the speed and/or acceleration without explicitly considering vehicle dynamics or any form of ADS framework. Specifically, Mena-Oreja et al. (2018) developed an open-source platooning simulator (i.e., PERMIT) based on SUMO to verify the effectiveness of platooning in terms of fuel reduction and congestion improvement. Though simulation data provide convincing results, the PERMIT platform considered traffic level control but did not consider vehicle dynamics or a proper ADS planning module. Secondly, most ADS studies that are based on vehicle-level simulators, such as CARLA (Dosovitskiy et al., 2017), over-simplify the perception and planning module, thus directly loading simulation results and generating waypoints, limiting the software's scope to a prototype algorithm. Moreover, due to the computational limitations,

though providing sufficient platoon-related results, studies that adopted CARLA (Hidalgo et al., 2021; Taylor et al., 2022) do not consider traffic level performance and introduce no background traffic. Lastly, the scope of conventional CACC studies (Guo and Ma, 2020; Milanés and Shladover, 2014, Porfyri et al., 2018) is limited to longitudinal control without explicitly considering any lateral movements among the ADS vehicles. Despite the definition difference, though the "cut-in" condition (i.e., randomly merged human-driven vehicle) is considered, the scope of Milanés (Milanés and Shladover, 2014) work still falls in longitudinal control but with additional consideration to damped the disturbance.

Accordingly, this study proposes a multi-lane platooning algorithm with a complete two-step (i.e., strategic mission planning and tactical motion planning) software framework. In particular, we define platooning as an organized behavior whereby each vehicle in a platoon has a responsibility towards the rest of the platoon to abide by certain agreed-upon rules. Regarding ADS structure, the algorithm utilizes perception results and composes a mission plan and a corresponding high-resolution motion plan that is executable for ADS control modules.

The focus of the proposed work is twofold, one of which is on "organized behavior." With platooning, a group of vehicles, coordinated by the platoon leader, aim to move through traffic as safely and efficiently as possible. In contrast, with CACC, independent vehicles only receive front vehicles' real-time information, establishing a longitudinal, feed-forward ad hoc string (i.e., a vehicle string where each member considers solely its preceding vehicle's status) that allows reduced following gaps. Once a platoon has formed, or the members have subsequently joined, they must follow a certain set of rules and protocols that the leader commonly sets. A platoon leader may set the rules simply by passing what was received from infrastructure (e.g., the maximum amount of vehicles allowed in a platoon or speed limit), whereas other rules may be set by the leader of his own volition (e.g., protocols for allowing vehicles to join a platoon). Additionally, because platooning is a group activity, cooperation among each member is allowed in situations when the leader would coordinate the relevant members to accomplish a more involving task, such as lane-change, on-ramp merge, and a cut-in join. It is important to emphasize such distinction, as most conventional studies use CACC and platooning interchangeably, regardless of these fundamental differences. Table 1 lists the differences between platooning and CACC.

The other focus is on adhering to the existing SOTA ADS platform's framework that harnesses onboard computation power and sensor suites (i.e., camera, Lidar, GPS, and DSRC) to establish end-to-end ADS vehicle automation and communication. Existing SOTA ADS platforms, such as CARMA℠ (Soleimaniamiri et al., 2021), Autoware™ (Kato et al., 2018), and Apollo™ (Fan et al., 2018), share similar ideologies in which the ADS tasks are accomplished by three sequential modules – perception, planning, and control. As one of the planning applications, the multi-lane platooning algorithm relies on the perception module (i.e., localization and V2V communication) to acquire the relative speed and position of the surrounding ADS vehicles, by selecting proper maneuvers based on the heuristic finite state machine (FSM) in the mission planning step. Afterward, the corresponding mission plan containing the start-to-end position and speed is fitted into a high-resolution motion plan (i.e., executable trajectory) that is ready to be executed by the control module. Notably, the proposed multi-lane platooning algorithm is parallel with other planning applications (i.e., lane cruising, lane change, etc.) and compatible with upstream and downstream ADS modules.

In the remainder of this paper, the multi-lane platooning algorithm is introduced in the methodology section. The performance of the proposed multi-lane platooning algorithm is then evaluated by using SUMO (Lopez et al., 2018) and CARLA (Dosovitskiy et al., 2017) simulators. The results are analyzed and further discussed. Finally, conclusions are provided along with future research directions.

2. Methodology

This study develops an SOTA ADS platform compatible multi-lane platooning algorithm that allows C-ADS-equipped vehicles to seek platooning opportunities in both the current lane and adjacent lanes. An SOTA ADS platform refers to a software system implemented in a highly automated C-ADS-equipped vehicle which provides and supports basic highly automated functions such as lane following and pure pursuit (i.e., pursuing a set target). It consists of three main levels (i.e., modules): perception level, planning level, and control level. The SOTA ADS platform also supports additional applications to expand its capabilities, such as emergency stop, work zone identification and avoidance, and emergency pull-over. The proposed multi-lane platooning algorithm is developed as one of the applications to perform in parallel with existing basic applications to provide the host C-ADS vehicle with multi-lane cooperative platooning capability.

The multi-lane platooning algorithm consists of two steps: mission planning and motion planning. The mission planning regulates the strategic plans for the host vehicle based on an FSM for organized behaviors. Notably, a genetic fuzzy system (GFS) intelligent controller is trained as a special treatment to handle the complex on-ramp merging scenario. Motion planning is the immediate succeeding step of mission planning, which generates a series of waypoints based on the appropriate strategic plan. We use a basic PID controller in this study to track the planned trajectory as the lower-level control is not the focus of this paper, but any other trajectory

Table 1
Differences between platooning and cooperative adaptive cruise control.

Category	Platooning	Cooperative Adaptive Cruise Control
Control hierarchy	Hierarchical control with special responsibilities for platoon leader.	Decentralized control with no special responsibilities for the string leader.
Membership	Coordinated platoon/group membership.	Ad hoc string membership and vehicle behave independently.
Spatial scope	Operates in a single or multiple lanes for a platoon lane change.	Operates in a single lane with small following gaps.

tracking method can be applied.

In this section, the two-step framework is introduced first. Then, the platooning behavior is presented. Specifically, a deterministic FSM was developed to guide the platooning operation in the Formation, Operation, and Dissolve superstates and regulate C-ADS-equipped vehicle behavior. Notably, the GFS is also included in this section as a secondary intelligent controller that operates in parallel to the FSM to handle complex scenarios such as the cooperative merge, which the FSM cannot handle well. Lastly, the trajectory generation algorithm is described to generate trajectories for C-ADS-equipped vehicles to complete specific types of behavior planned by the platooning behavior protocol. The proposed multi-lane platooning algorithm distinguishes itself from the previous ones, because:

- The multi-lane platooning algorithm abide by the protocols of the modern ADS software stacks, interacting with existing perception and control modules and operating in parallel with other ADS applications to fully deploy ADS functions on C-ADS-equipped vehicles.
- The multi-lane platooning algorithm covers decision making at different levels. i.e., strategic mission level and tactical motion level.
- The algorithm handles platooning under complex driving scenarios on multi-lane highways by using a combination of rule-based and learning-based methods.
- The algorithm uses trajectory generation and sharing for completing various behavior types such that planned trajectories of other relevant C-ADS-equipped vehicles can be fully considered (i.e., intent sharing, of predictive nature).

2.1. Algorithm framework

As one of the planning applications, the proposed multi-lane platooning algorithm operates in parallel with all other ADS planning applications, e.g., ACC, inline cruising, intersection, and work zone. Specifically, each application follows the two-step framework (i.e., strategic mission and tactical motion planning) in which the mission planning leverages the perception result to find the optimum strategic plan while the motion planning generates a corresponding trajectory as a list of waypoints. The arbitration process of electing the proper planning application is one of the main functionalities of the SOTA ADS platforms. We will use the terms "mission planning" and "motion planning" for simplicity in Fig. 1 and in the remainder of the paper.

Fig. 1. shows the logic flow of the multi-lane platooning algorithm in the two-step framework. The first step is mission planning, whereby one of the pre-defined semantic mission plans is selected based on the current scenario. The mission plan corresponds to the identified scenario and recommends a proper maneuver based on the deterministic FSM as high-level semantic information (e.g., lane cruise, lane change, on-ramp merge, etc.) representing the ADS strategy of the next planning horizon. The maneuver plan is categorized into multiple pre-defined plan types, whereby each type contains essential information (e.g., start/target location, start/target speed, and start/target lane, etc.) for the motion planning step. Notably, the principles of the deterministic FSM involved in the mission planning step will be introduced in the next session. Secondly, the motion planning step parses the information from the maneuver plan, generating a detailed trajectory connecting the current position to the desired position following the target speed. The trajectory plan is valid through the current planning window, but it will be updated on a higher frequency to prevent the false maneuvers caused by obsolete information. The trajectory is in the form of a series of connected waypoints, whereby each waypoint consists of the target position, target speed for this specific location, and target angle for this specific location. Such data structure allows the control module to follow each waypoint with the desired speed and angle by manipulating the steering, throttle, and brake. The trajectory generation algorithm, which is presented in detail in the following sections, manages the behavior of the host vehicle by taking as input the desired

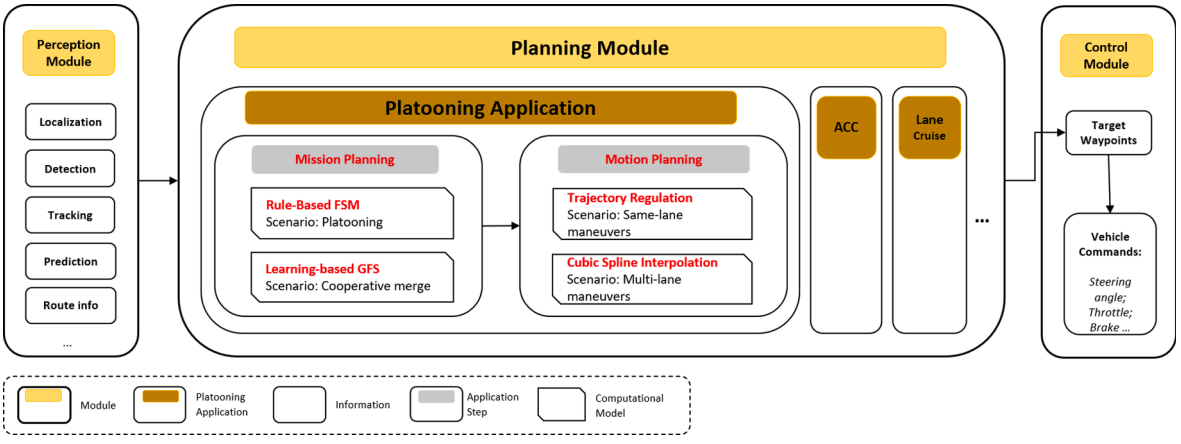


Fig. 1. © 2022 University of California, Los Angeles (UCLA) Mobility Lab.
Logic and data flow of the platooning algorithm.

ADS maneuvers and the desired intra-platoon gap as input, thus providing an executable trajectory plan for the downstream control module.

Additional effort was conducted in terms of algorithm deployment. ADS operation involves transmitting V2V data and processing related modules' results. To this end, we implemented a dedicated messaging system to subscribe to the existing module's result and publish necessary information (i.e., behavioral decisions and planned trajectories) generated by the proposed algorithm. To further improve communication efficiency, we regulated the message as status messages for all platoon members versus request-response messages which are dedicated to related C-ADS-equipped vehicles. This process significantly simplifies the communication overhead. On the other hand, interaction with downstream modules requires accurate and efficient calculation. Most existing vehicle stringing (such as CACC) algorithms focus purely on traffic performance, whereas computational accuracy and efficiency are usually overlooked. We addressed this research gap by utilizing the rule-based FSM strategic planner and the frenet (Crenshaw et al., 1993) coordination system for behavioral organizations. The rule-based FSM operates with linear complexity that allows efficient strategic decision-making. The frenet (Crenshaw et al., 1993) coordination system uses the down-track and cross-track distance to express relative position in regard to the host vehicle. The system is constantly rotating as an observer moves along the cure. Hence it is always non-inertial (i.e., a frame system that undergoes acceleration with respect to an inertial frame system) and provides better accuracy than cartesian coordinate system, especially when calculating relative positions.

2.2. Behavior protocol

A deterministic FSM is designed to manage the platooning process with three superstates, as shown in Fig. 2, such that the platooning behavior is organized. Each of the three superstates—Formation, Operation, and Dissolve and each of these superstates is composed of several states with their internal logic to guide the platooning process.

A C-ADS-equipped vehicle can be either controlled by a human driver (during human operations) or ADS applications (during automated driving). If the platooning plugin of the C-ADS-equipped vehicle is enabled, it will engage in the Formation superstate and seek platooning opportunities by negotiating with surrounding platoon leaders. After joining a platoon, the ego C-ADS-equipped vehicle will switch to the Operation superstate to regulate its speed and keep the desired time gap and/or headway. Once a platoon member intends to leave the platoon, it will request to leave and switch to the Dissolve superstate after getting permission from the platoon leader. A C-ADS-equipped vehicle can keep seeking platooning opportunities after leaving the previous platoon, or it can

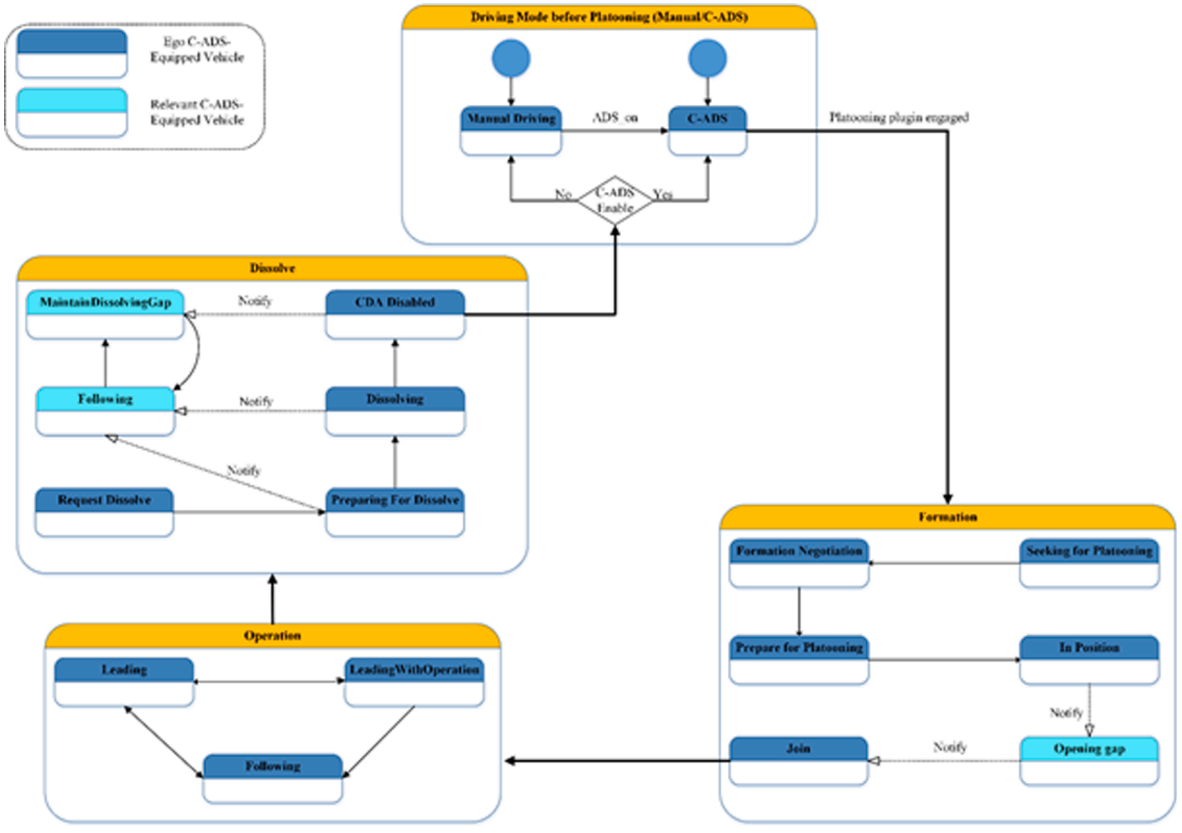


Fig. 2. © 2022 UCLA Mobility Lab.
The Finite State Machine for platooning.

disable the platooning plugin and switch to manual driving.

2.3. Formation

The Formation superstate consists of six states: Seeking for Platooning, Formation Negotiation, Preparing for Platooning, In Position, Opening Gap, and Join. As shown in Fig. 3, a state can transition to another state based on the given events or notifications, or it can continue the same state to complete the behavior. The detail of each state and its transitions are introduced in this section.

Seeking for Platooning state is an initial state during which the single C-ADS-equipped vehicle communicates with surrounding platoon leaders and shares its formation intent. The C-ADS-equipped vehicle will automatically enter this state after it enables the platooning plugin. There are two transitions in the Seeking for Platooning state:

- Self-transition: Maintain the Seeking for Platooning state if there is no candidate platoon
- To Formation Negotiation state: At least one candidate platoon or a single platooning-enabled C-ADS-equipped vehicle is found

2.4. Operation

As shown in Fig. 4, the *Operation* superstate consists of three states: *Leading*, *Leading with Operation*, and *Following*. The C-ADS-equipped vehicle will switch to the *Operation* superstate after joining a platoon. The detail of each state and transitions are introduced in this section.

The first vehicle of a platoon is the leader of this platoon, and the platoon leader will exclusively possess the *Leading* state among all platoon members. The platoon leader implements a hierarchical control and is responsible for coordinating all maneuvers between members and interactions with vehicles outside the platoon. There are three transitions in the *Leading* state:

- Self-transition: Maintain the Leading state and regulate its speed to keep a safe distance to the preceding vehicle (i.e., one that is not part of the platoon) or keep the desired speed
- To Leading with Operation state: If the platoon leader achieved a multi-lane formation agreement with a C-ADS-equipped vehicle
- To Following state: If the leader becomes a follower (e.g., a separate single C-ADS-equipped vehicle performs a front-join)

2.5. Dissolve

As shown in Fig. 5, the Dissolve superstate consists of five states: *Request Dissolve*, *Preparing for Dissolve*, *Dissolving*, *Maintain Dissolving Gap*, and *CDA Disabled*. A platoon member will request permission from the platoon leader before the dissolve process. The detail of each state and transitions are introduced in this section.

If a platoon member, including the leader, intends to leave the current platoon, it will switch to the *Dissolve* state and send a request to the platoon leader to initiate the *Dissolve* process. There are two transitions in the Dissolve state:

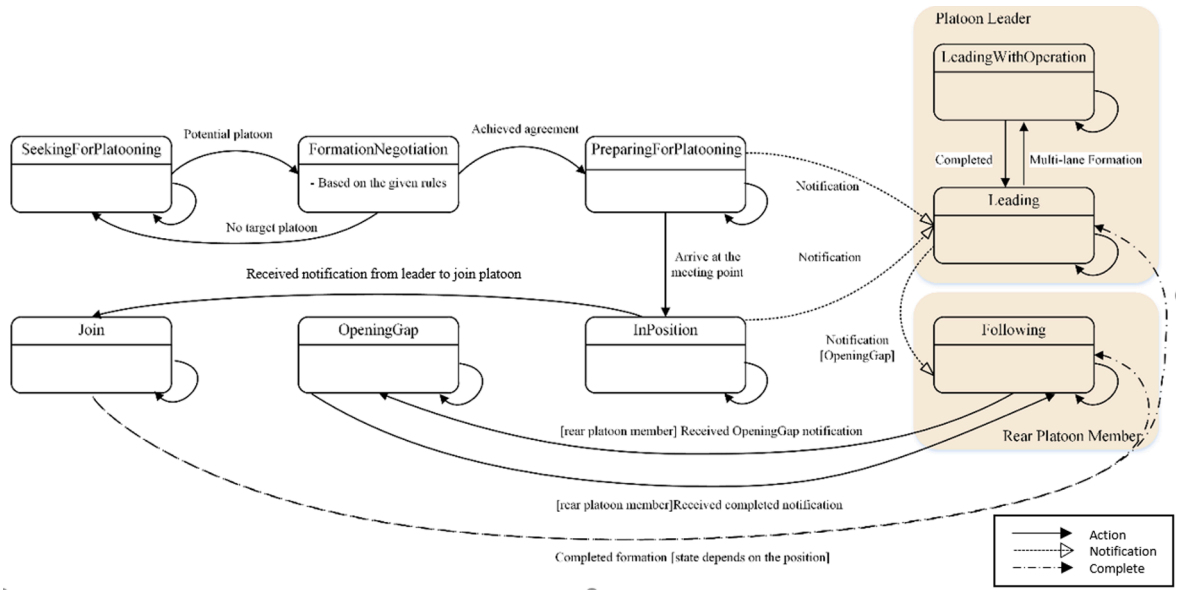


Fig. 3. © 2022 UCLA Mobility Lab.
FSM for the Formation superstate.

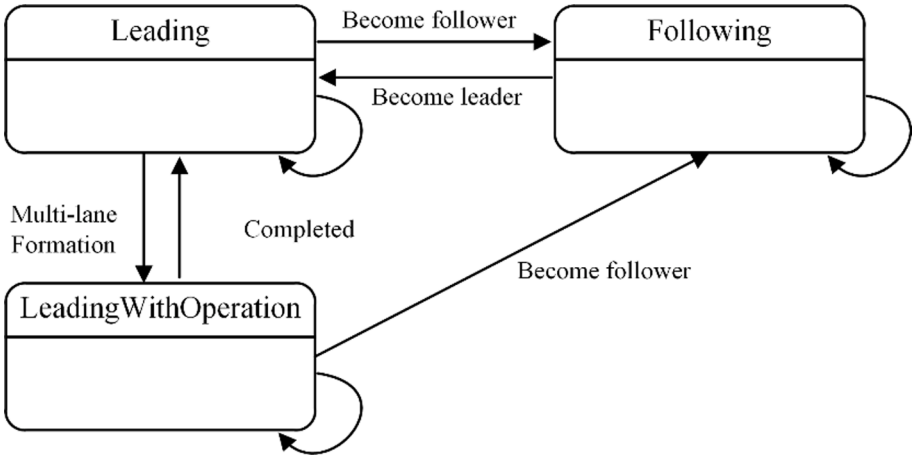


Fig. 4. © 2022 UCLA Mobility Lab.
Finite State Machine for the Operation Super-state.

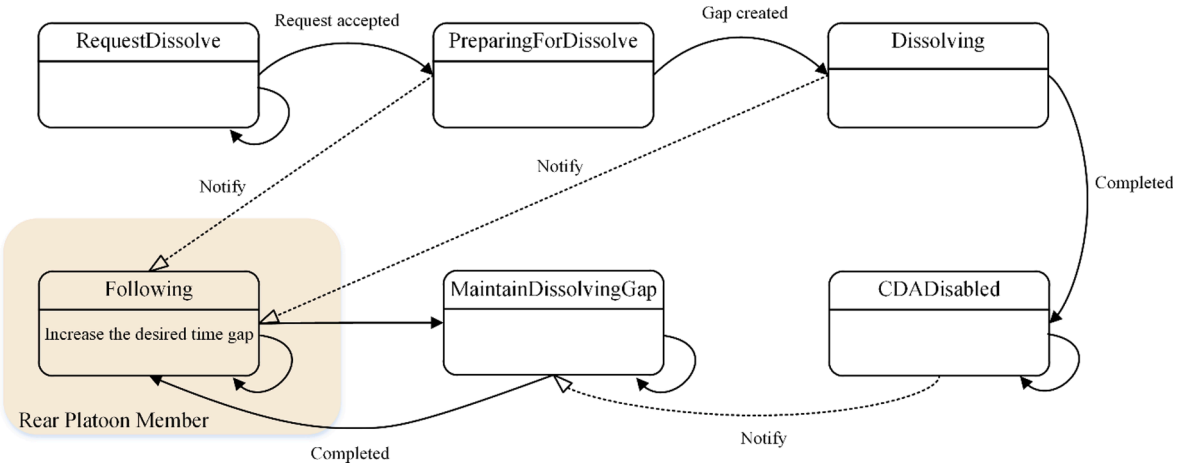


Fig. 5. © 2022 UCLA Mobility Lab.
Finite State Machine for the Dissolve Super-state.

- Self-transition: If the dissolve request is denied
- To Preparing for Dissolve state: If the dissolve request is accepted

2.6. Intelligent GFS controller

Due to the excessive number of traffic merging scenarios, with and without platoons, conventional deterministic rule-based systems may suffer from two perspectives: Rule specification can become extremely difficult and the number of rules will explode. Therefore, a GFS is adopted to handle the complex cooperative merge scenarios. Generally, in a GFS, a Genetic Algorithm (GA) trains a set of parameters of a fuzzy logic system, which includes membership functions of all inputs, outputs, and the rules used to define the input–output relationship (Sathyan et al., 2021). In the proposed algorithm, the inputs information of GFS are the position, speed, and target lane of the surrounding six vehicles (front and rear vehicles from the same lane, left lane, and right lane), whereas the outputs are the desired position, desired speed, and target platoon index for the host vehicle. In V2V on-ramp merging scenarios, the GFS performs as an off-line trained, intelligent controller alongside the existing rule-based FSM. At the same time, the entire system relies mainly on the rule-based controller (i.e., FSM) during all other situations due to its computational efficiency and robustness (i.e., no training is needed and applicable regardless of network geometry). Two cases are considered with the emphasis on maintaining robust cooperation among mainline and merge lane C-ADS vehicles, i.e., merging with and without pre-formed mainline platoons.

2.6.1. Vehicle-Vehicle merging

As part of the proposed multi-lane platooning algorithm, each C-ADS-equipped vehicle in the network is equipped with the same

intelligent GFS controller which is capable of making lane change decisions and controlling its speed, acceleration, and lane change behavior in a decentralized manner. That is, each C-ADS-equipped vehicle only uses local information available from its nearby vehicles (obtained via sensor or V2V communications) to make decisions. This decentralized behavior is trained without the existence of a platoon and serves as the baseline controller to enable individual ADS behaviors, which lays the foundation for vehicle-platoon cooperation. The cooperative single-vehicle ADS GFS is referred to as the GFS-M model, where M stands for "merge" in later sections. Note that during a cooperative merge operation, both the merging vehicle and the mainline gap vehicle are controlled by the GFS model due to the vehicle-vehicle cooperative nature of the algorithm.

The GFS controller uses 21 inputs to manage lane change and acceleration. These inputs include the distance, speed, and signals (blinkers and brake lights) of the closest vehicle ahead and behind the C-ADS-equipped vehicle moving on the left, current and right lanes of the ego C-ADS-equipped vehicle. The inputs to the GFS also include ego vehicle positions (i.e., x, y coordinates) and speed. The recommended acceleration and lane change are the outputs from the GFS. The acceleration is a continuous variable, whereas the lane change is a discrete variable with three classes: (1) move right, (2) stay on the lane, and (3) move left. In terms of training, since the strategies that cause inefficient merge will reduce the average speed, we formalize a fitness function that maximizes the average speed of all vehicles in the network with the constraints of a smooth acceleration and safety distance. Each training episode has a duration of 5 minutes, with actions that control speed and lane change only taken during intervals of 0.5 seconds. This interval is used only during training to keep the training time reasonable and can be reduced during testing. As a result, the GFS was trained to cooperate with other single ADS vehicles with the same model on board. The detailed training method is introduced in the following section.

2.6.2. *Training process*

The schematic of the training process is shown in Fig. 6. We adopt the Fuzzy Bolt framework (Sathyan et al., 2021) that trains a single GFS model to reduce the search space effectively. A GA is initialized with random individuals that define the population to train the proposed GFS. The GFS is set up for training with ten triangular membership functions for each input. In terms of output, five triangular membership functions are used to calculate the continuous acceleration output. Three membership functions are adopted to define the three classes of the lane change maneuvers, i.e., lane change to left, lane change to the right, and stay in the current lane. Each GA individual is a vector consisting of the parameters related to the GFS controller, which includes the boundaries of the membership functions and the rules in the rule-base. Thus, each individual in the GA can be converted to a GFS controller, which can then be used to simulate an episode (e.g., in SUMO). An episodic fitness can be evaluated for each such individual in the GA. The fitness would represent the system's ability to achieve the objective of smooth merging. GA individuals with higher fitness values are more likely to be selected for crossover and mutation, thus getting selected for the next generation. In contrast, individuals with lower fitness scores have a higher chance of elimination from the population. This process of evolution continues for a pre-defined number of generations or until convergence, where the fitness score has little to no improvement (i.e., lower than 0.01) for 20 iterations. Each episode is a scenario simulated for a timeframe of 5 minutes with actions taken at intervals of 0.5 seconds. The episode starts at $t = 120$ seconds and ends at $t = 420$ seconds, where the initial 120 seconds warm-up period populates the traffic density before training. The GFSs evolve with each generation during training to improve the fitness function. At the end of each generation, the best model of the generation is applied to a validation episode to check its effectiveness on an unseen scenario.

The training maximizes the fitness function, which captures the average speed of all vehicles across the 5-minute episode. For efficiency, a particular stopping criterion is designed to stop episodes that result in undesired cases, such as stopped vehicles in the network. This function allows early truncation of undesirable episodes, avoiding meaningless episodes in an early stage, thus improving the training time. The GFS controllers that result in such situations are penalized heavily. Thus, the fitness function that is to

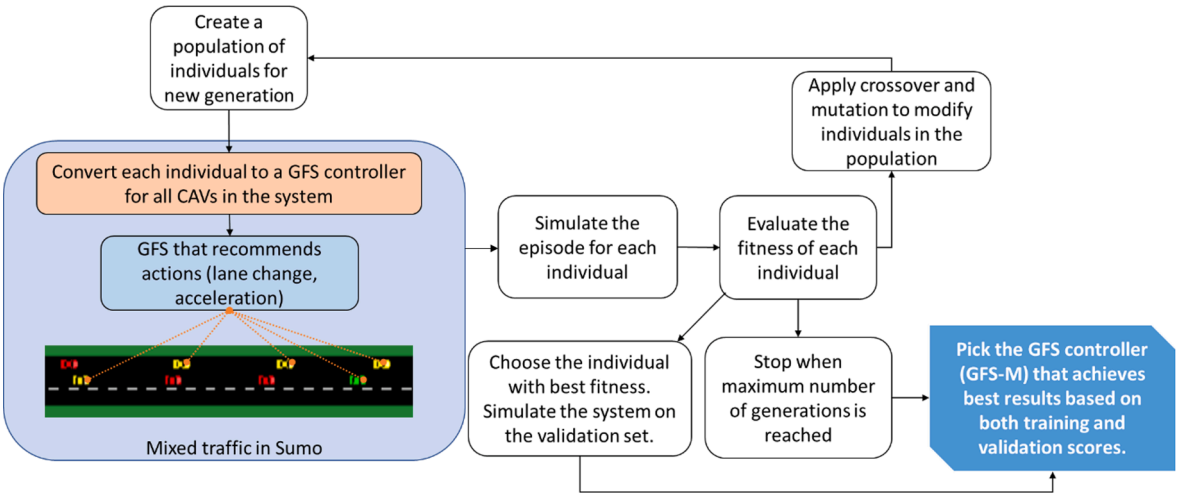


Fig. 6. © 2022 UCLA Mobility Lab.
Training Process of GFS.

be maximized by GA algorithm is defined as follows.

$$F = \begin{cases} 0.001 t_{end} if\, t_{end} < 420s \\ mean(v) if\, t_{end} = 420s \end{cases} \quad (1)$$

The top portion of equation (1) relates to early stoppage scenarios in an episode before the final time t = 420 seconds. These individuals in GA are heavily penalized. Ideally, the simulations should run for the entire time till t = 420 seconds, in which case, the average speed of all the vehicles should be maximized, as shown in the bottom portion of equation (1).

2.6.3. Vehicle-platoon merging

The GFS also considers cases where C-ADS-equipped vehicles merge onto a mainline containing vehicle platoons. The platoons follow the deterministic FSM mentioned in Section 2.2. Similar to the single-vehicle case, the GFS for vehicle-platoon cooperation is trained separately in combination with the FSM protocols. Because the platoon establishes hierarchical control architecture, the leaders are trained to allow cooperative merge when a merging ADS vehicle occurs. The behavior of the follower vehicles in a platoon is always coordinated by the platoon leader, as noted in the hierarchical platoon protocol. Other C-ADS-equipped vehicles that engage with platoon leaders to join platoons will also follow platoon protocol. The individual C-ADS-equipped vehicles that are not part of a platoon or have not engaged with any platoons are controlled by the previously trained GFS-M model.

The vehicle-platoon scenario poses some additional challenges to the cooperative merge behavior. The merging vehicle has to work cooperatively with the platoon leader to merge into the platoon in the mainline for different situations, as shown in Fig. 7. The platoon leader needs to make decisions on speeding up, slowing down, or allowing the merging vehicle to perform a cut-in join at a position as defined in the Section 2.2, platooning protocol.

Similarly, the FuzzyBolt framework (Sathyan et al., 2021) is used to train the GFS model that controls the platoon leaders. The GFS model that controls platoon leaders is named GFS-PL. The training schematic is shown in Fig. 8. The GFS-M model from the previous training controls the individual vehicles that have not engaged with any platoons. We used the same fitness function defined in the previous sections because the objective of a cooperative merge with platoons includes maintaining the smooth flow of traffic.

In this case, only the GFS-PL module is trained. This GFS-PL module contains two GFS sub-models: (1) the GFS-PL-speed submodel for controlling the speed of the platoon leader (GFS-PL-speed) and (2) the GFS-PL-score submodel that predicts a performance score (i.e., considering comfort, safety, and network congestion) for all available merging positions. The two submodels are trained simultaneously. Once the merge position is determined, the rear vehicle relative to the merge switches to OpeningGap, and the GFS-M starts to control its speed. Simultaneously, the speed of the merging vehicle is controlled to align it horizontally with the merge position in the platoon. Once the merging vehicle is between the two relative platoon members corresponding to the merge position, the GFS-M controls the speed to allow the host vehicle to merge into the platoon. For instances where an individual C-ADS-equipped vehicle on the mainline encounters a merging vehicle, the GFS-PL will control the mainline vehicle to decide the behavior of the vehicles to allow

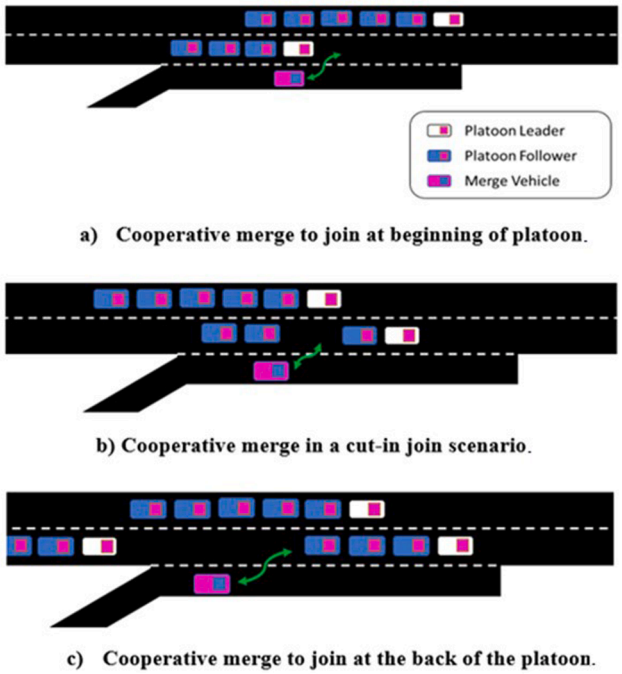


Fig. 7. © 2022 UCLA Mobility Lab.
Cooperative Merge to join at the back of the platoon.

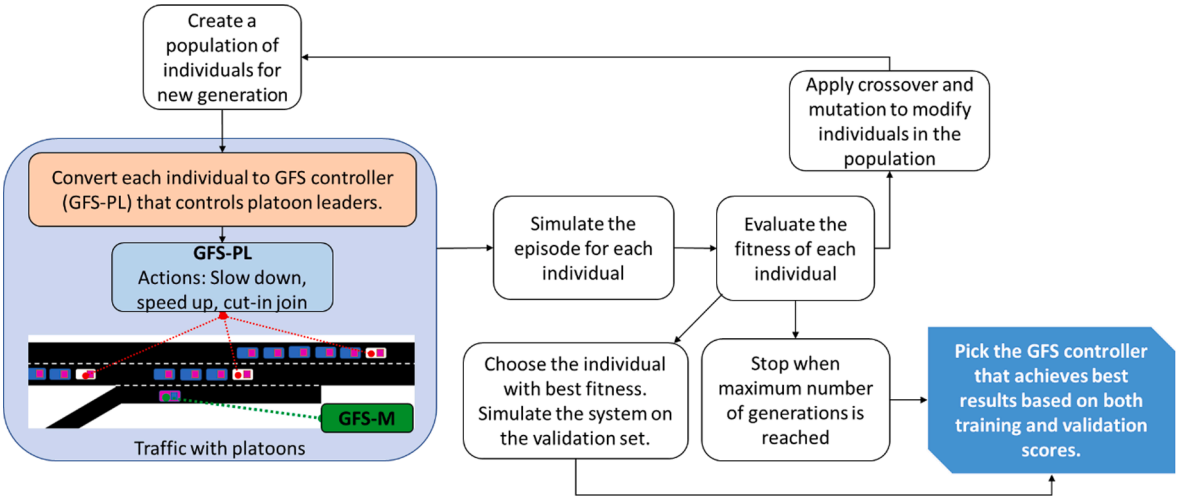


Fig. 8. © 2022 UCLA Mobility Lab.
Training Process for GFS-PL module.

them to form a two-vehicle platoon.

2.7. Trajectory generation

2.7.1. General maneuver trajectories

The multi-lane platooning algorithm generates behaviors through the deterministic FSM (i.e., mission planning step), and then trajectories are generated to complete each of the behaviors (i.e., motion planning step). Trajectory generation is a core module connecting high-level decision-making on behavior and lower-level control that outputs throttle and brake levels. Trajectories are usually regenerated based on a real-time dynamic world model and need to be smooth to avoid system disutility, such as excessive jerks to ensure driving comfort. Many types of trajectories (e.g., polynomials, splines, etc.) have been used in real SOTA ADS platforms. In this study, we adopt a widely used method, i.e., the cubic spline interpolation, to generate the trajectories for different maneuvers. Other methods can be used to flexibly replace the spline-based approach.

In a general ADS implementation, when a global route is generated by the global route planner using the A-star algorithm (Russell,

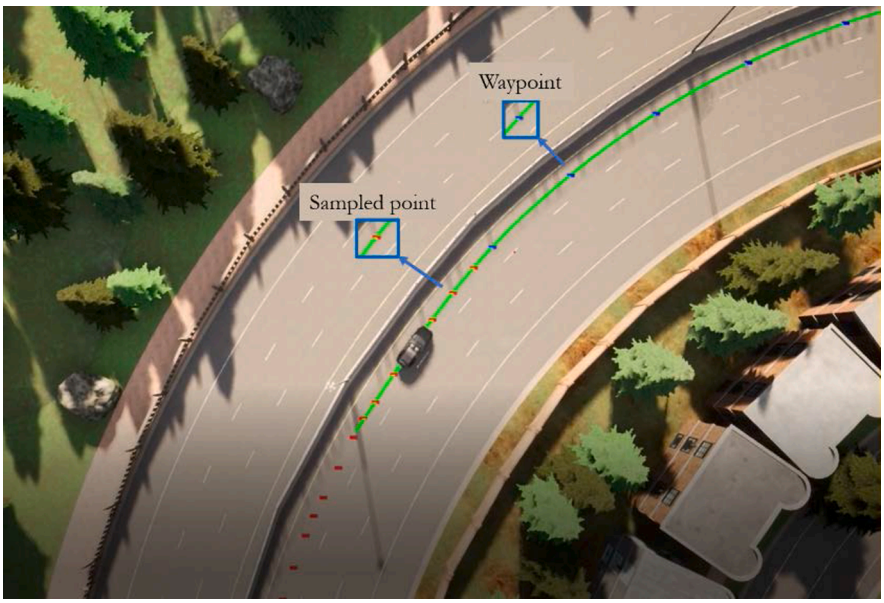


Fig. 9. © 2022 UCLA Mobility Lab.
Illustration of splines (square red dot), route history (curved green line), waypoints, and sampled points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2010), a series of waypoints can be generated to lead the vehicle to the destination, and a pre-specified distance separates each nearby pair of points (e.g., 6 m), as shown in Fig. 9 where the generated waypoints along the route are highlighted as square blue dots. However, a fixed separation distance between two consecutive waypoints leads to unstable controls due to the various radius of the road curvature. To this end, according to the ego vehicle's location, the proposed algorithm generates a smooth curve among the following N waypoints (e.g., N= 8) by using cubic spline interpolation. As shown in Fig. 9, the highlighted green line represent the generated smooth curve which forms an interim continuous trajectory in-between waypoints, resulting in a smooth trajectory that minimizes the unstable movements and, in contrast to other forms such as polynomials, connects all of the waypoints within the calculation range (i.e., the pre-defined N points). Note that the vehicle will travel along the green line (i.e., cubic spline) as long as the current cubic spline remains valid.

Once the interim continuous cubic spline is generated, it is discretized as a series of candidate sample points with a specific spline resolution (i.e., spl_{res}). In particular, this study assigns the resolution to maintain a 0.1 s time gap between candidate sample points. The ego vehicle then samples discrete trajectory points from the candidate sample points based on the current location, speed, and target speed (i.e., target speed at the next waypoint). The related derivation is shown in equation (2) – (7), and red dots represent the sampled trajectory points in Fig. 9. Specifically, equation (2) calculates the spline resolution based on the current and target speed; equation (3) finds the number of candidate points that satisfies the target spline resolution; equation (4) finds the index of the candidate sample points while equation (5) finds the x-coordinates of each candidate sample point; equation (6) finds the y coordinate of the candidate sampled points; and equation (7) defines the cubic spline where A, B, C, and D, are cubic spline parameters to be determined.

$$spl_{res} = avg(spd, spd_{tar}) \times t_{res} \tag{2}$$

$$n_{spl} = \left\lfloor \frac{dis}{spl_{res}} \right\rfloor \tag{3}$$

$$idx_i = \left\lfloor \frac{i \times spl_{res}}{dis_{res}} \right\rfloor, i = 1, 2, \cdots, n_{spl} \tag{4}$$

$$x_i = X(idx_i) \tag{5}$$

$$y_i = Y(idx_i) \tag{6}$$

$$Y = AX^3 + BX^2 + CX + D \tag{7}$$

Where:
spl_{res} = sample resolution (distance).
spd = current speed of the ego vehicle (distance/time).
spd_{tar} = target speed at the next waypoint (distance/time).
$avg(spd, spd_{tar})$ = average of current speed and target speed at the next waypoint (distance/time).
t_{res} = time resolution (0.1 s in this study).
n_{spl} = number of candidate sample points along the continuous cubic spline.
dis = distance to the next waypoint or to a specified location (which is usually represented by a waypoint).
i = the sequence of the sampled point.
idx_i = index of the sampled point.i
dis_{res} = the distance between two consecutive candidate sample points, i.e., candidate sample point resolution (0.1 m in this study).
X = *the* x-coordinates of the interpolation points between waypoints, and each of them has a unit distance of 0.1 *meters*.
Y = y-coordinates of candidate sample points, and A, B, C, and D are determined by cubic spline interpolation method.
x_i = x-coordinate of the sampled point.i
y_i = y-coordinate of the sampled point.i

Specifically, the discretized trajectory only covers the two nearest waypoints (i.e., the current waypoint and the consecutive waypoint). Once the ego vehicle reaches the later waypoint (i.e., distance smaller than 0.5 meters), its interim planned continuous trajectory are updated to cover the upcoming N waypoints (e.g., N= 8) on the route, and the discretized trajectory between the next two consecutive waypoints is re-sampled and updated by repeating the same process. This process guarantees that the vehicle's yaw angle is aligned with the upcoming route, as the discretized trajectory is always a segment of a continuous spline. Note that the number of waypoints used for cubic spline calculation is configurable while the current setting for the study (i.e., $\boldsymbol{N} = 8$) follows the default value in CARLA, the simulator used in the case study.

2.7.2. Platooning regulation trajectories

The above-mentioned trajectory generation process is the general process that applies to any isolated ADS-equipped vehicles as well as platoon leaders when the leadersare in lane following mode. The speed and acceleration of the platoon members (i.e., all followers) usually need to be regulated against the immediately preceding vehicles (i.e., gap regulation) and/or the platoon leader (i.e., headway regulation). The gap and regulation processes are usually combined to ensure platooning performance such as stability.

During a gap regulation, a platoon follower needs to consider its desired time gap and the preceding vehicle's position during the trajectory generation, as shown in equation (8)

$$pos_j^g(t) = \frac{pos_{j-1}(t) - L_{j-1} + pos_j(t-dt) \times \frac{gap_d}{dt}}{1 + \frac{gap_d}{dt}} \tag{8}$$

Where:

$pos_i(t)$ = the position of the vehicle j at time.t

$pos_{j-1}(t)$ = the position of the preceding vehicle j −1 at time.t

L_{j-1} = the length of preceding vehicle.j −1

dt = time resolution, i.e., simulation step (0.1 s in this study).

$pos_j(t-dt)$ = the position of the vehicle j at last time step.t −dt

gap_d = desired time gap of the vehicle.j

During a headway regulation, we can also replace the desired time gap with the time headway by following equation (9), but the target vehicle is the platoon leader.

$$pos_j^h(t) = \frac{pos_1(t) + pos_j(t-dt) \times \frac{h_d}{dt}}{1 + \frac{h_d}{dt}} \tag{9}$$

Where:

h_d = desired time headway of the ego vehicle j to the platoon leader.

In this study, the ego follower vehicle is regulated by a leader-predecessor scheme by following equation (10)

$$pos_j(t) = \mu \bullet pos_j^g(t) + \lambda \bullet pos_j^h(t) \tag{10}$$

$$\mu + \lambda = 1 \mu > 0, \lambda > 0$$

Where:

$pos_j(t)$: the planned position of the Leader-Predecessor regulation.

μ: the gain on the gap regulation value.

λ: the gain on the headway regulation value.

Note that equation (10) will be applied for each member/follower of the platoon, starting from the second vehicle, the predecessor of which is the platoon leader. Since the leader's trajectory has been planned through the trajectory generation process by considering the surrounding traffic environment, the trajectory of each of the followers can be calculated subsequently. This process distinguishes this regulation method with other reactive ones (i.e., it is a trajectory-based regulation of a predictive nature), but also seamlessly integrates well with SOTA ADS platforms that require planned trajectories.

3. Results

The simulation evaluation in the study requires perspectives from both vehicle/automated driving and traffic management. Regular traffic or automated driving simulators cannot meet this requirement. Therefore, a novel co-simulation tool, OpenCDA, which the authors developed for CDA research, is adopted (Xu et al., 2021). OpenCDA is a simulation tool integrated with a prototype cooperative driving automation (CDA; see SAE J3216) (SAE International, 2020) pipeline as well as regular automated driving components (e.g., perception, localization, planning, control). The tool integrates automated driving simulation CARLA (Dosovitskiy et al., 2017), traffic simulation SUMO (Lopez et al., 2018), and Co-simulation (CARLA + SUMO). Although SUMO can well test the behavior protocol by using simple ADS behavior (i.e., car-following and lane change behavior models), CARLA is needed to test detailed ADS and C-ADS vehicle behavior that is controlled by ADS and C-ADS software stack. One critical aspect of this study is the trajectory generation and control components of C-ADS vehicles and the possibility of using vehicle dynamics models to understand C-ADS vehicle behavior as controlled by the software pipeline. Therefore, the simulation evaluation is formularized as follows. (See OpenCDA (Xu et al., 2021) for a detailed discussion of simulator selection and applications):

- SUMO simulation evaluation: SUMO was used to tune the proposed multi-lane platooning algorithm, observe the platooning behaviors in various MPRs, and train the GFS controller in multiple use cases (i.e., vehicle-vehicle merging and vehicle-platoon merging).
- The focus here is to fully consider platooning effects on large-scale traffic. Therefore, a traffic-levels simulator such as SUMO is necessary. The simulator ensures that the algorithmic parameter, particularly those parameters at the mission planning level, is optimized or trained to improve traffic system performance, which exceeds the scope of the conventional platooning studies that only focus on platooned vehicle performance.
- CARLA simulation evaluation: CARLA was used to test detailed vehicle behavior controlled by the proposed platooning protocol becasue CARLA can simulate the full pipeline of an SOTA ADS software platform and some level of vehicle dynamics.
- CARLA + SUMO co-simulation: The co-simulation was adopted to test the algorithm performance comprehensively in a complex multi-lane highway scenario with large-scale background traffic. Specifically, SUMO was adopted to spawn and control the background traffic with realistic behaviors (i.e., NGSIM (Punzo et al., 2011) trajectory or calibrated IDM (Kesting et al., 2010)), while the C-ADS vehicles run the proposed multi-lane platooning algorithm in the CARLA simulator. The two simulators operate in synchronous mode, so vehicles in both simulators can achieve two-way interaction in each step. In this way, we can observe the

detailed platooning and merging vehicle behavior (via CARLA) and create realistic traffic scenarios to test the platooning operations (via SUMO).

3.1. Platooning traffic simulation in SUMO

Though the proposed multi-lane platooning algorithm is based on organized behavior, several parameters at both mission planning and motion planning were introduced throughout the pipeline to adjust the algorithmic performance. In addition, the GFS controller needs to be trained before deployment. In this section, we tune and evaluate the multi-lane platooning algorithm in SUMO, demonstrating its effectiveness while the GFS controller is trained in various scenarios in SUMO. The section is divided into four sections: parameter tuning with synthetic trajectory, single-lane platooning simulation in SUMO, capacity analysis at different MPRs, and GFS training.

3.1.1. Parameter tuning

Before simulating and evaluating the performance of the multi-lane platooning algorithm, several parameters need to be determined. The inter-platoon and intra-platoon were determined based on sensitivity analyses in previous studies (Soleimaniamiri et al., 2021; Kato et al., 2018). Therefore, the other three parameters, including negotiation distance, time-gap for join, and time-gap for Dissolve are tuned by the GA.

Based on the nature of those three parameters, value encoding is used where each gene of the the GA algorithm is represented by a real number with two decimal places. The population size is 50 and the generation is 100 for seeking the globally optimal combination of the three parameters. To prevent any premature convergence, the two-point crossover is applied with a crossover rate of 0.9, and the mutation rate is set to 0.1. The Roulette Wheel selection is utilized to select potential optimal solutions in the parental generation. The total delay is utilized as the fitness value. The searching range of each parameter is as follows:

- Negotiation distance: 0 $\sim$ 60 *meters*
- Time-gap for Join: 0.6 $\sim$ 1.5 seconds
- Time-gap for Dissolve: 0.6 $\sim$ 1.5 seconds

Convergence of the GA Tunning Process (Fig. 10) shows the convergence of the GA tuning process. The solid blue line indicates the average fitness of all chromosomes, whereas the solid orange line represents the best fitness of each generation. It can be found that the tuning process is terminated after 72 generations as the fitness score converges. The optimized parameters are listed below:

- Negotiation distance: 53.20 *meters*
- Time-gap for Join: 1.14 seconds
- Time-gap for Dissolve: 0.96 seconds

Since the simulation resolution in this study is not less than 0.1 s, the value of the two time-gaps are rounded to one decimal, i.e., the time gap for Join is set to 1.1 s, and the time gap for Dissovle is set to 1 s.

The control gains of the gap regulation and headway regulation in equation (10) are pre-specified in this study and tuned to find a better combination of the μ and λ. Because the μ is defined as $\mu = 1 - \lambda$, the adjustment range of μ is set to 0.6 to 1 using increments of 0.1 to investigate the performance. Other values less than 0.6 will not be investigated based on the initial simulations and evaluations because the platoon followers cannot maintain the desired time gap during the entire trip.

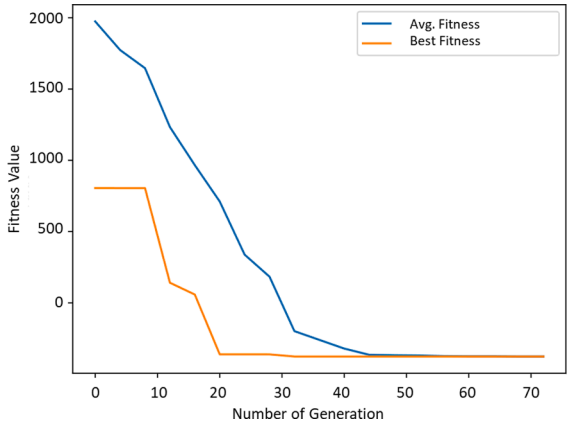


Fig. 10. © 2022 UCLA Mobility Lab.
Convergence of the GA Tuning Process.

3.1.2. Single-lane platooning simulation in SUMO

A synthetic trajectory is generated to simulate stop-and-go traffic. The leading vehicle is an HDV (id: 0), and it follows the given trajectory. A platoon is regulated by the multi-lane platooning algorithm with different values of μ. When $\mu \leq 0.8$, it is difficult for the platoon followers to maintain the desired intra-platoon time gap during the operation; when $\mu = 1.0$, some platoon members cannot maintain the standstill with the stop-and-go traffic. Therefore, the $\mu = 0.9$ (as shown in Fig. 11) is utilized in this study by considering the performances under different traffic environments.

3.1.3. The capacity of basic freeway segment at different MPRs

The pipeline capacities (i.e., the capacity of basic freeway segments) of different C-ADS-equipped vehicle market penetration rates (MPRs) were investigated with the tuned parameters. The MPR varies from 0 % to 100 % with a 20 % increment. The freeway segment is 2800-meters long with two lanes. The first 200 m at the beginning of the freeway only allow the same-lane formation to form stable platoons. The speed limit of this freeway segment is 113 kilometers/hour (70 miles per hour), and the acceleration of C-ADS-equipped vehicles is limited to 1 m/s^2 for comfortable driving behavior. Based on previous studies, the sensor detection range is set to 120 meters.

Both CACC (Milanés and Shladover, 2014; Liu et al., 2018) and the proposed multi-lane platooning algorithm are investigated, as shown in Table 2. Generally, the capacity increases with the growing MPR, from 1972 vehicles per hour per lane (vphpl) to 3296 vphpl, by implementing either CACC or the multi-lane platooning algorithm.

As shown in Table 2, at 20 % and 40 % MPRs, the pipeline capacity of CACC is close to (slightly higher than) the platooning. For one, with low MPR, the C-ADS-equipped vehicles are sparsely located in mixed traffic, leading to fewer cooperation opportunities and formation agreements. Therefore, the stop-and-go oscillation is hardly reduced with limited formed platoons under the HDV-dominant traffic flow. In addition, the main reason for the low reduction in oscillation is that the multi-lane platoon formation behavior can slightly impact the upstream traffic at low market penetrations rate due to the lack of cooperation between HDV and C-ADS-equipped vehicles. For example, suppose a single C-ADS-equipped vehicle reaches an agreement with the target platoon leader and implements a cut-in join. In this case, the rear platoon members will slow down to create a safe gap for the single ego C-ADS-equipped vehicle; therefore, the upstream traffic, especially the HDVs, will slow down. Once the single ego C-ADS-equipped vehicle completes the lane change, the rear platoon member will lead all upstream platoon members to accelerate to maintain the desired intra-platoon gap. Though there is no significant delay for C-ADS-equipped vehicles in this process, HDV traffic from upstream may be negatively impacted due to the platoon formation speed changes. Since the HDVs are predominant in the traffic stream at low MPRs, this impact is significant, and therefore the pipeline capacity is negatively affected to a small extent. Note that under the low MPRs (up to 40 %), the capacity difference between CACC and the multi-lane platooning algorithm increases with the growing MPR. This increase occurs because more multi-lane formation opportunities will present with the growing MPR, and therefore more oscillations can be generated associated with the multi-lane formations.

Under moderate and high MPRs, i.e., when the C-ADS-equipped vehicles are predominant in the traffic stream, the multi-lane platooning algorithm can improve the pipeline capacity by up to 3.1 %. As previously mentioned in this section (i.e., Section

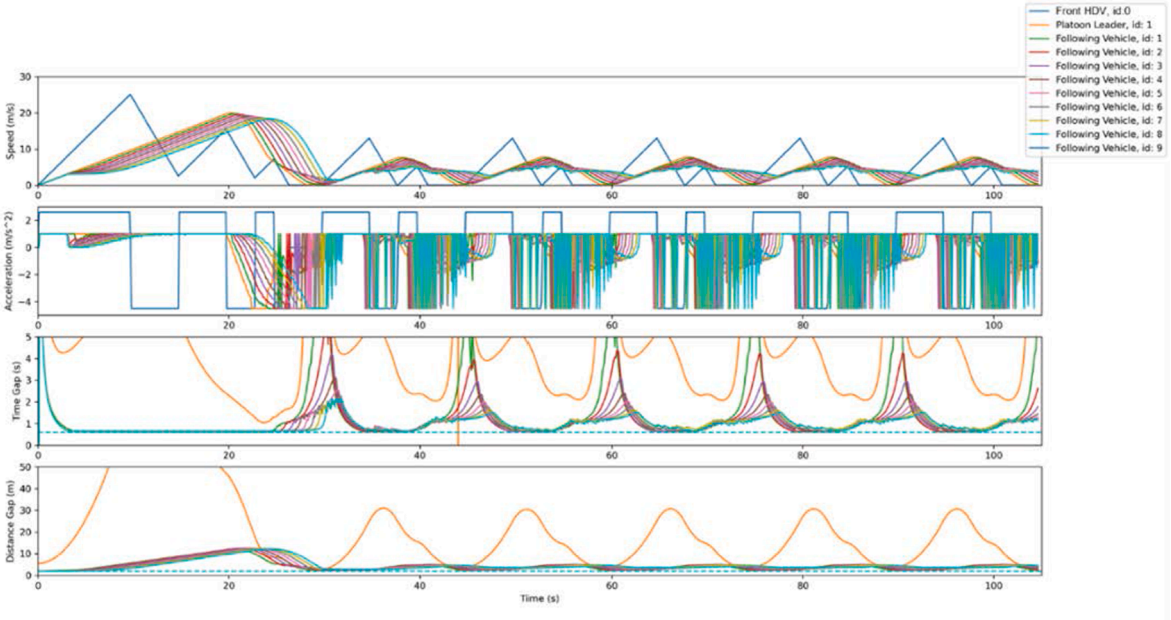
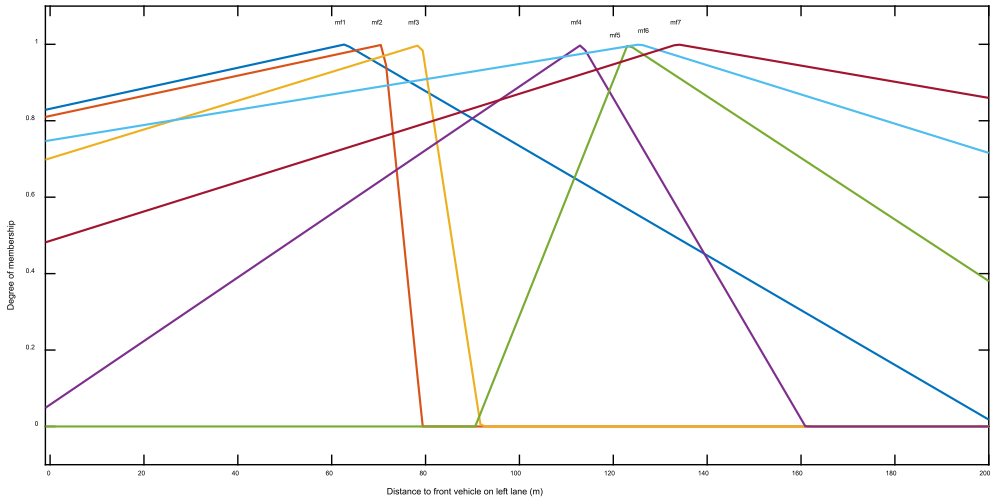


Fig. 11. © 2022 UCLA Mobility Lab.
Speed, Acceleration, Time Gap, and Distance Gap Results with Different Values of..$\mu = 0.9$

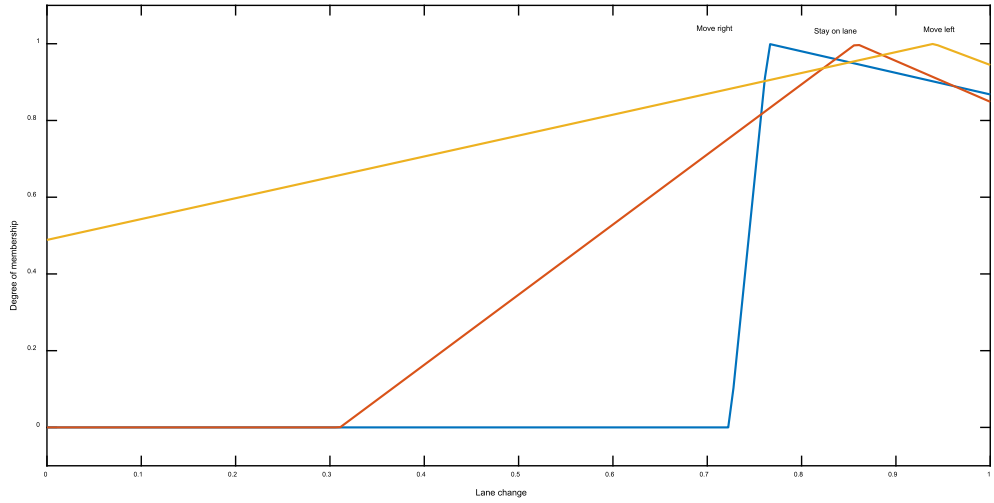
Table 2
The capacity of CACC and the multi-lane platooning algorithm.

MPR (%)	CACC (vphpl)	Platooning (vphpl)	Difference (%)
0	1972	1972	0.0
20	2080	2063	−0.8
40	2256	2212	−2.0
60	2588	2644	2.2
80	2808	2896	3.1
100	3296	3296	0.0

3.1.3), the C-ADS-equipped vehicle can be well controlled to recover soon from multi-lane join, and the impact will diminish with the increasing MPR. Also, the multi-lane formation provides the opportunity for the single C-ADS-equipped vehicle to utilize the reduced time gap and regulate its speed with the more stable regulation algorithm instead of using ACC (Milanés and Shladover, 2014). It is worth noting that under 100 % MPR, there are no lane changes with high traffic volume inputs because C-ADS-equipped vehicles can form platoons with others in the same lane. Therefore, the capacities of CACC and Platooning are the same under 100 % MPR.



(a) Membership function of GFS-M input after training



(b) Membership function of GFS-M model output after training

Fig. 12. © 2022 UCLA Mobility Lab.
Input and Output Membership functions of the GFS-M model.

Note that even though the capacity enhancements of platooning over CACC are only limited, the multi-lane platooning algorithm can benefit platoon members and the traffic from other aspects. For example, when negotiating with the platoon leader, the single C-ADS-equipped vehicle will choose the platoon with the same or similar destination. This choice will limit unnecessary formations and dissolves on longer freeway segments with on-ramps and off-ramps, thus reducing traffic disturbances. Reducing this type of human-made disturbance can help to improve capacity and reduce potential safety risks (Guo and Ma, 2020). Notably, the comparison listed in Table 2 is conducted only in conventional highway segments that are identical to the CACC experiment environments. In non-basic segments where lane changes are a necessity, the proposed multi-lane platooning algorithm is the superior solution because of its complex cooperation capability and trajectory-sharing (i.e., intent-sharing) nature.

3.1.4. GFS training for cooperative-merge

As aforementioned in Section 2.6.3, a GFS controller was adopted to handle the on-ramp merge situation due to the excessive number of scenarios and significant speed difference between mainline and merge lane vehicles. As merge lane C-ADS vehicles approach the merging point, the cooperation may be established as vehicle-vehicle merging or vehicle-platoon merging. Accordingly, we formulate the training in two steps. A GFS-M vehicle-vehicle merging controller was trained during the first step, whereas the vehicle-platoon merging controller, in which the controller from the first step was used during the training, was trained in the second step.

The training involves two scenarios for both steps. To be exact, scenario one contains 2000 vphpl mainline volume and 800 vphpl merge ramp volume, reaching 4800 vph total volume combined for two lanes and one merge lane. In contrast, scenario two populates 1500 vphpl mainline volume, 1500 vphpl merge ramp volume, and the total volume adds up to 4500 vph. Considering the random arrival of vehicles and the usage of different random seeds, the simulation can realize many different combinations of traffic density. As a result, the trained GFS-M model handles traffic scenarios around medium to high densities, allowing efficient vehicle-vehicle cooperative merging that minimizes the total system delay.

The membership functions of the trained GFS-M model for one of its inputs, distance to the front vehicle on the left lane, is shown in Fig. 12(a). Note that only the membership functions that were picked in the trained model are shown in the figure. At the start of the training, each variable was assigned 10 random membership functions, while the trained model used only 7 membership functions for this input variable. Similarly, Fig. 12(b) shows three membership functions that define the acceleration. Since acceleration is a continuous value, centroid defuzzification is used where all the rules are evaluated and aggregated using a weighted average to obtain the final acceleration value. Meanwhile, the discrete lane change output is obtained via mean-of-max defuzzification, from which an output class is obtained based on the output membership function defined for the most significant rule.

The trained GFS-M was evaluated on both scenarios. The improved throughput and delay under different MPRs are shown in Fig. 13. As more trained C-ADS vehicles are added to the network, the flow of traffic becomes smoother, improving the throughput and reducing the average delay.

Next, the second step of training in which each C-ADS vehicle is simulated using the trained GFS-M model as a baseline was conducted. The GFS-PL model was trained to optimize the speed of the platoon leader and the merge position for vehicle-platoon cooperation. Fig. 14. shows the validation result of the trained GFS-PL model in both scenarios, where the throughput increases

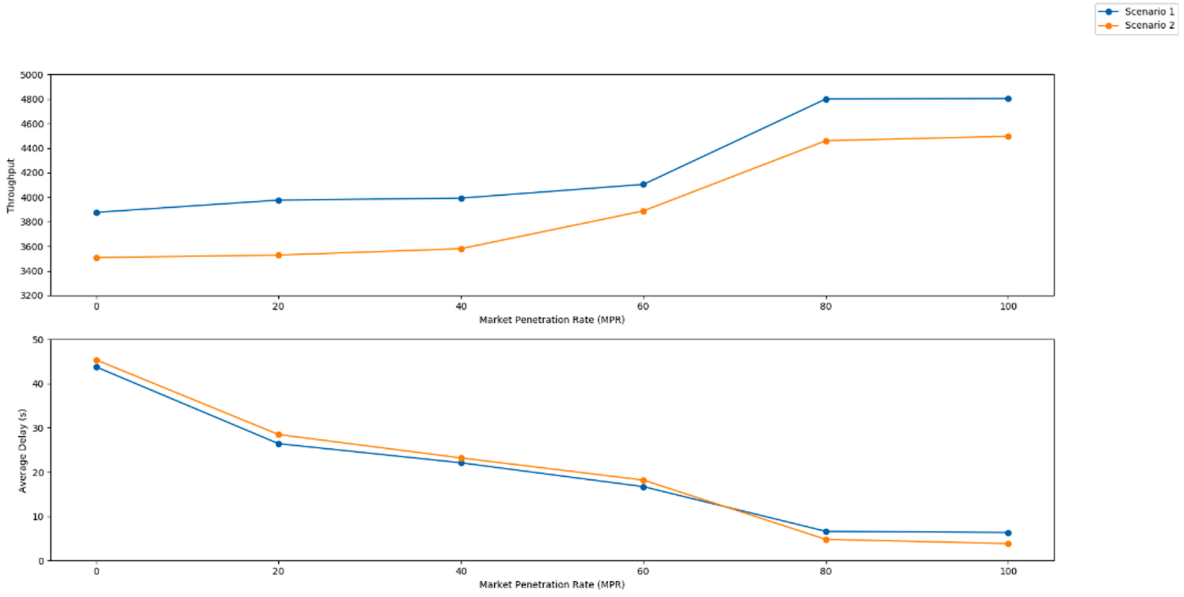


Fig. 13. © 2022 UCLA Mobility Lab.
GFS-M validation results (throughput and average delay) for different MPRs.

and the average delay decreases with the increase in MPR. Overall, this trend is similar to the cooperative vehicle-vehicle merge (as shown in Fig. 13). However, the throughput for scenario one reaches the maximum network capacity of 4800 vph much earlier than vehicle-vehicle merging, at an MPR of 40 %, because the merge lane volume is low and the disruption to the main traffic can be handled more easily with vehicle-platoon cooperation.

In contrast, the merge density in scenario two was considerably higher (1500 vph). Therefore, for lower MPR, where cooperation by HDVs is not assured like the C-ADS vehicles, the throughput is lower, and the time delay is higher. The network throughput with MPR and reaches a maximum when the MPR reaches 80 %. The increase in MPR of C-ADS vehicles will coordinate merge volume better with the mainline traffic and gradually increase the capacity.

3.2. Platooning simulation in CARLA

Due to its microscopic nature, SUMO cannot simulate vehicular dynamics, including acceleration, deceleration, and jerk. However, as an ADS planning application, it is essential for the proposed algorithm to integrate with downstream control modules. In particular, the proposed algorithm incorporates dynamic vehicle models, physics models, and a trajectory generation model to interact with the downstream control models by providing an executable trajectory for the next planning horizon.

Therefore, to demonstrate such integration with downstream control modules, we implemented CARLA as the simulator to conduct vehicle-level simulation evaluations, as described in this subsection. The evaluation is divided into three sections: platooning with synthetic trajectory, platooning with NGSIM trajectory, and cooperative merging.

3.2.1. Platooning simulation in CARLA with synthetic trajectory

This scenario tests the platoon's stability, which is indicated by the degree of amplified oscillations when the leading vehicle changes speed dramatically. In detail, a five-vehicle platoon keeps driving in the same lane while the platoon leader follows a design speed profile to accelerate and decelerate frequently. This scenario was designed because we aimed to evaluate the algorithm's capability to maintain the desired intra-platoon time gap in CARLA.

The target intra-platoon time gap was set as 0.6 s in the synthetic cycle testing. As one example in Fig. 15 demonstrates, the platoon followers were able to keep the designed time gap of 0.6 s during the whole process, even with the leading vehicle dramatically increasing and decreasing speeds. When the platoon leader started to accelerate suddenly, the platoon members were able to follow it tightly without any speed-overshooting. When the platoon leader rapidly stepped on the brakes, the followers were able to smoothly decelerate at a comfortable rate and stay within constant time gaps between each other, which indicates the stability of the platooning.

3.2.2. Platooning simulation in CARLA with NGSIM trajectories

In the real trajectory testing, we select many challenging trajectories from NGSIM datasets to understand the platoon following behavior. In the examples in Fig. 16, as vehicles in CARLA are launching from a standstill, it takes about 5 s until the leader is close enough to the HDV in front of it to start using its car following algorithm. Once the leader is engaged in following the HDV, despite the HDV's frequent speed changes, the platoon leader is able to follow it securely and smoothly. In particular, for both NGSIM trajectories,

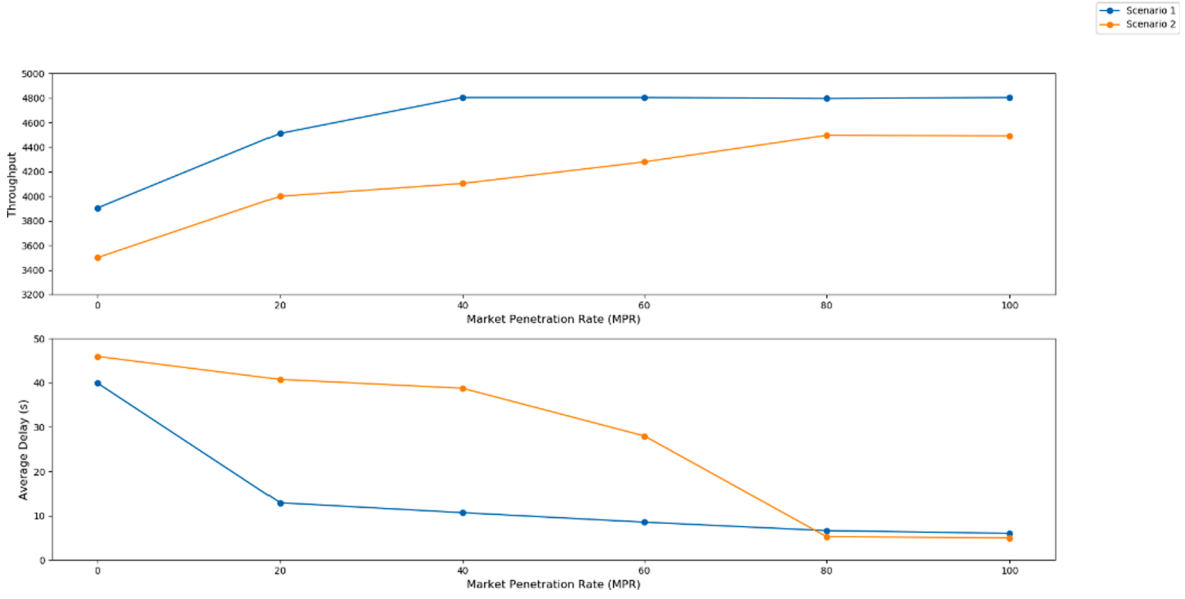


Fig. 14. © 2022 UCLA Mobility Lab.
GFS-PL validation results (throughput and average delay) for different MPRs.

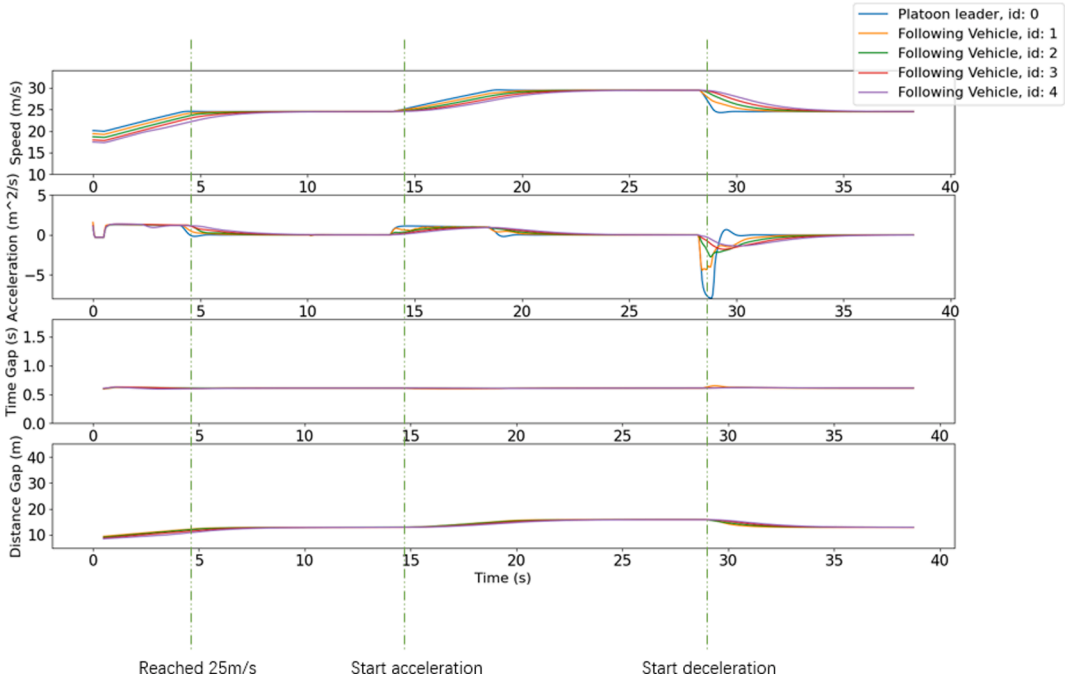


Fig. 15. © 2022 UCLA Mobility Lab.
Synthetic trajectory results.

a 1.5 second time gap was successfully maintained throughout the platoon, with damped acceleration disturbances originating from the real-world trajectory. At the same time, the platoon members can keep a constant time gap of 0.6 s even when the front HDV rapidly accelerates or decelerates, indicating the robustness of the proposed algorithm in a real-world setting.

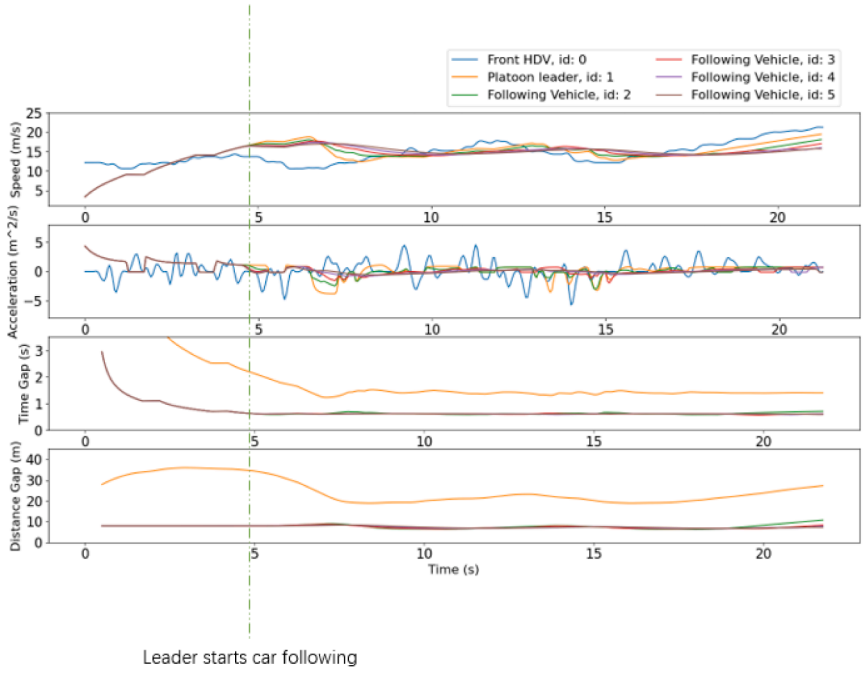
3.3. Cooperative merge in CARLA-SUMO co-simulation

As one of the ADS applications, the proposed algorithm provides seamless integration with upstream perception modules and the downstream control modules. The existence of HDV background traffic is crucial to demonstrate such integration. Specifically, the platoon establishes different behaviors when the detected HDV is in front, on the side, or in a different lane. During operation, the platoon leader is expected to maintain a steady gap between the preceding HDV while monitoring the HDVs on the side to avoid crashes. During merging, the C-ADS vehicles should detect mainline vehicle types and select the target platoon accordingly. These organized behaviors constitute crucial testing scenarios for integration validation with upstream perception modules, as they are not only part of the proposed multi-lane platooning algorithm and they depend heavily on the perception results.

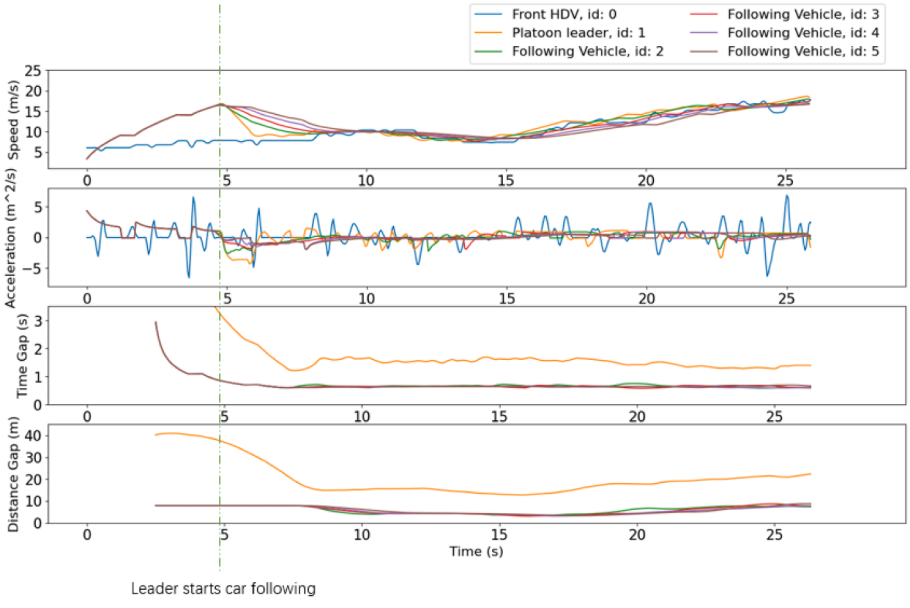
CARLA provides vehicle dynamics simulation and roadway environment. However, it falls short when offering large-scale HDV background traffic with realistic (i.e., calibrated IDM (Kesting et al., 2010)) driving profiles, which are essential to test the algorithm's behavior with the existence of the HDV traffic stream. SUMO, on the other hand, supports more realistic and larger-scale background traffic simulations with significantly less computational cost. Therefore, a CARLA-SUMO co-simulation was conducted with maximum C-ADS vehicle dynamics and realistic background traffic behavior for cooperative merge scenarios. Fig. 17 is a snippet of a platooning test that was performed under the co-simulation setting.

CARLA and SUMO operate in a server-client mechanism where the simulation operates on a local Internet Protocol (IP) address with a dedicated Transmission Control Protocol (TCP) host address. Running both simulators at the same IP address while setting the corresponding port address in both simulators is required to establish communication between the two. Once the connection is established, CARLA-SUMO co-simulation requires an identical simulation map on both simulators. Though CARLA runs in a much more realistic environment, the basic traffic network between the two simulators has the same opendrive format (Dupuis, 2010). Therefore, it is important to edit the basic traffic network in opendrive format and share this file between the two simulators. With the opendrive file, the CARLA simulator can populate the detailed road surface and environment, whereas SUMO simply interprets the network as a node-edge structure. During testing, the co-simulation requires using synchronous mode for both simulators. This setting limits the simulation clock speed, which is based on the slowest port at each step to ensure the simulation on both simulators is synced. As a result, each vehicle's stepwise information is shared across simulators, though the calculation was performed on different simulation simulators.

We also developed a heuristic joining method (HJM), which serves as the baseline merge algorithm to be compared to the GFS controller. The HJM is a sorting-based method that chooses one platoon member with the shortest Euclidean distance for the merging



(a) NGSIM trajectory 1



(b) NGSIM trajectory 2

Fig. 16. © 2022 UCLA Mobility Lab.
NGSIM trajectory results.

vehicle as its future preceding member in the platoon. The HJM employs a practical approach that is not necessarily optimal but sufficient enough to reach the immediate goal.

For the co-simulation scenario, the mainline has a high-speed traffic flow mixed with human-driven vehicles managed by SUMO and the C-ADS-equipped vehicles that are controlled by CARLA. When the single C-ADS-equipped vehicle is near the merging area, it will communicate with the mainline platoon and request to join. Once the merging vehicle and platoon leader have achieved an agreement, the single C-ADS-equipped vehicle must finish the merge and join the platoon simultaneously before the acceleration lane

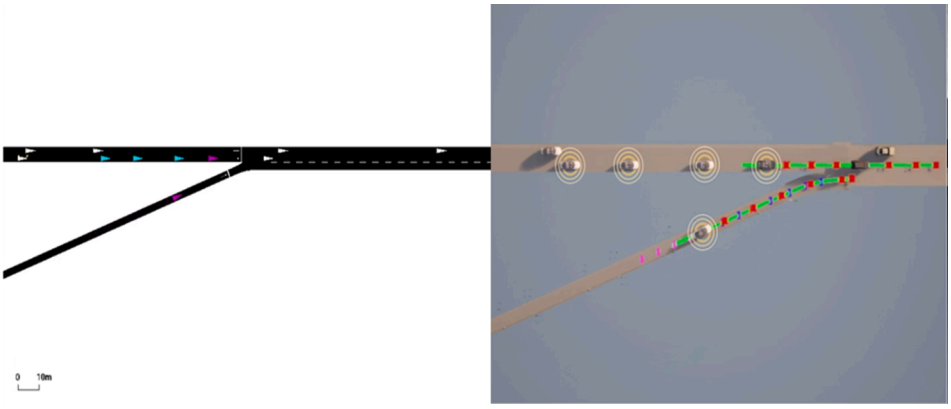


Fig. 17. © 2022 UCLA Mobility Lab.
A snippet of a platooning scenario test under the co-simulation setting. From left to right: Sample simulation snippet in SUMO, the corresponding view in CARLA where the green lines and red dots represent planned trajectory path and points, and the blue dots and pink dots represent on-route waypoints and historical trajectories, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

ends. Unlike a heuristic-based method, GFS considers platoon members' position, speeds, and surrounding human-driven vehicles' information, aiming to accomplish joining maneuvers while maintaining a globally optimized traffic delay. Therefore, the network level performance of GFS is expected to be superior, though the merging sequence may differ from the HJM's suggested result.

Fig. 18 shows the merging position from both HJM and GFS. HJM chooses the position between the original second and third platoon members as the best merging point, while GFS selects the position between the fifth and sixth platoon members as the merging point. Such difference is generated by the optimization goal of these two methods: HJM tries to reduce the joining time and thus selects the closest position, whereas GFS aims to minimize the traffic delay caused by the joining process and does not seek an immediate completion to the merge operation at the cost of overall traffic performance. Additionally, GFS's strategy reduces the abruptness of lane changes, which may cause upstream oscillations and leaves the vehicle enough space to speed up.

The results produced by these two different algorithms are shown in Fig. 19. As Fig. 19 **(a)** demonstrates, after the joining request is approved, HJM only takes 7 s to finish both the gap opening and the merging vehicle change lane. In contrast, it takes about13 s for GFS to finish the lane change since the merging vehicle needs to slow down to wait to join the gap behind it, as Fig. 19 **(b)** shows. Although HJM takes less time to finish the maneuver, potentially, it may affect safety, traffic delay, and energy assumption more negatively because the three platoon members (ID5, ID6, ID7) are required to reduce the speed dramatically. As Fig. 19 **(b)** demonstrates, compared withHJM, the cooperative merge influences only one platoon member (ID 8) under GFS's strategy. Notably, joining vehicle's following gap decreased in the 30 seconds after the joining started but before lane change occurred. This trend reflects the GFS's preference to speed up the joining vehicle before the lane change and then slow it down afterward to gradually increase the gap (i.e., slow down speed) to the pre-defined steady value. This preference reduces the merging gap created by the mainline platoon members. In comparison, the HJM method handles the lane change with a bigger merging gap but requires all downstream members to accelerate and close the gap after changing lanes, which causes more disturbance (i.e., stop-and-go behaviors, abrupt speed changes, etc.) and fuel use. As a result, the mainline platoon experienced minimal speed change and disturbance, yielding an improved throughput.

Overall, our results indicate that the GFS successfully learned, during the traffic environments via SUMO, to optimize the traffic impact through a complex nonlinear mapping between the inputs (such as the speed differences between the mainline and merging vehicles) and strategy outputs, in a process that is hard to capture by arbitrarily defined heuristic rules. Regarding large-scale traffic, such a joining preference will significantly reduce the cost in terms of traffic delay, safety and energy consumption.

4. Conclusion

CDA technologies have the potential to improve roadway capacity, travel reliability, and traffic performance. In the previous studies, a single-lane vehicle stringing algorithm, or CACC, was developed, whereas our study proposes a multi-lane platooning algorithm with organized behavior via a hierarchical control structure for more complicated and practical situations. Based on the simulation results, multiple key observations and implications are summarized as follows:

- The proposed multi-lane platooning algorithm can efficiently guide the behavior of platoon members and external C-ADS-equipped vehicles aiming to join or leave a platoon in adjacent lanes under each superstate and precisely switch between the superstates and states under different complicated scenarios.
- The combination of gains in gap regulation and headway regulation can significantly impact the capability of maintaining the desired time gap. The results indicate the importance of considering both the immediately preceding vehicle and the platooning leader's trajectories to maintain the desired time gap and standstill distance.

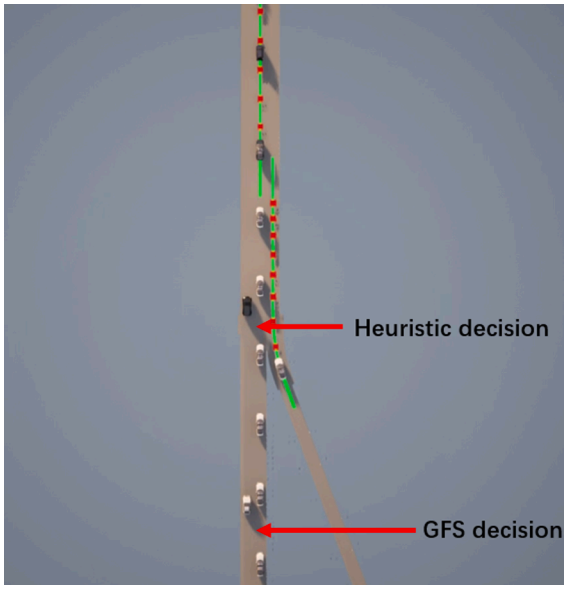


Fig. 18. © 2022 UCLA Mobility Lab.
Different decision makings between HJM and GFS.

- With intent/trajectory-sharing for CDA, C-ADS-equipped vehicles can be proactive about others' intentions, and the platooning Formation, Operation, and Dissolve superstates are more efficient and effective under increasing MPRs. Therefore, the proposed multi-lane platooning algorithm can improve traffic level performance further.
- The proposed algorithm adopts a GFS algorithm as a secondary controller to handle the more complex and hard-to-define cooperative merge scenarios. This unique structure compromises efficiency in pre-defined scenarios and accuracy in rare corner cases, demonstrating the capability of both rule-based and learning-based solutions.

Moreover, the proposed algorithm adheres to the existing SOTA ADS platforms, subscribing to the perception module as input and publishing the desired trajectory in a data structure compatible with the downstream control module. Within the multi-lane platooning algorithm, the mission planning step leverages the deterministic FSM to propose a suitable strategy for the current situation. The subsequent motion planning step transforms the desired mission plan into a series of waypoints using cubic spline interpolation that calculates a smooth trajectory to maintain vehicle stability. Overall, the proposed algorithm is a planning application that can be deployed in parallel with other planning applications on SOTA ADS platforms.

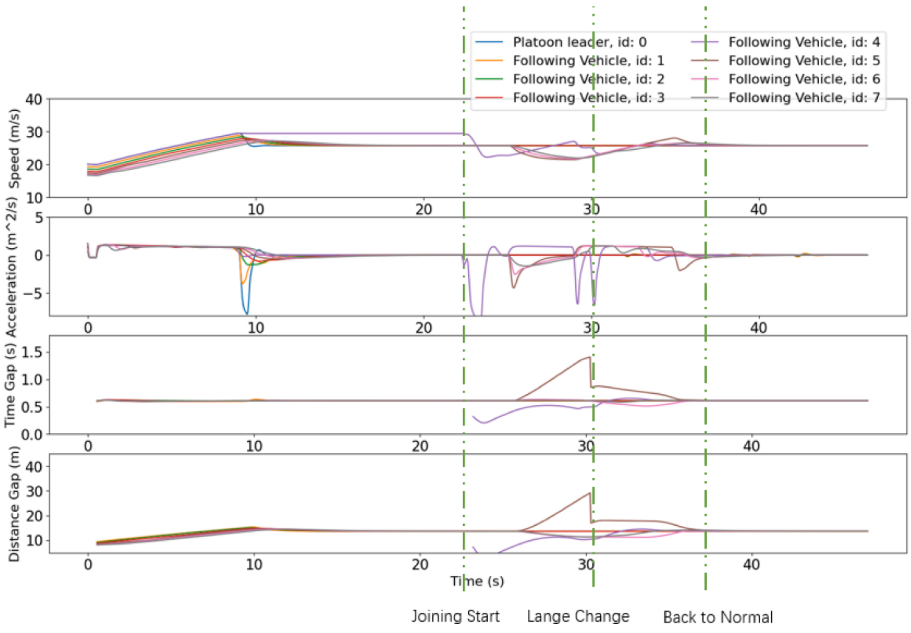
However, due to the natural construction of the leader–follower structure, accurate, efficient, and secure V2V communication is crucial to such applications. The experiments were conducted assuming that the V2V communication had no noise or delay, and the C-ADS vehicle has sufficient computation power to handle the communication and reasoning. In more realistic scenarios, algorithm performance under compromised communication capability and computational power are future research topics that need further discussion. Moreover, a stochastic version or real-time optimization and artificial intelligence/machine learning methods can be utilized in the future to improve the performance further by considering the local performance and the corridor traffic performance. On the other hand, the gains in gap regulation and headway regulation are fixed, leaving additional research gaps for dynamic and adaptive control under different scenarios. Lastly, the contribution of this study focuses on the innovations in the planning stack (at both mission and motion planning levels). Future studies can consider integrating the sensing and perception module with planning and control to explore end-to-end or semi-end-to-end solutions for cooperative platooning and merging.

Author contributions

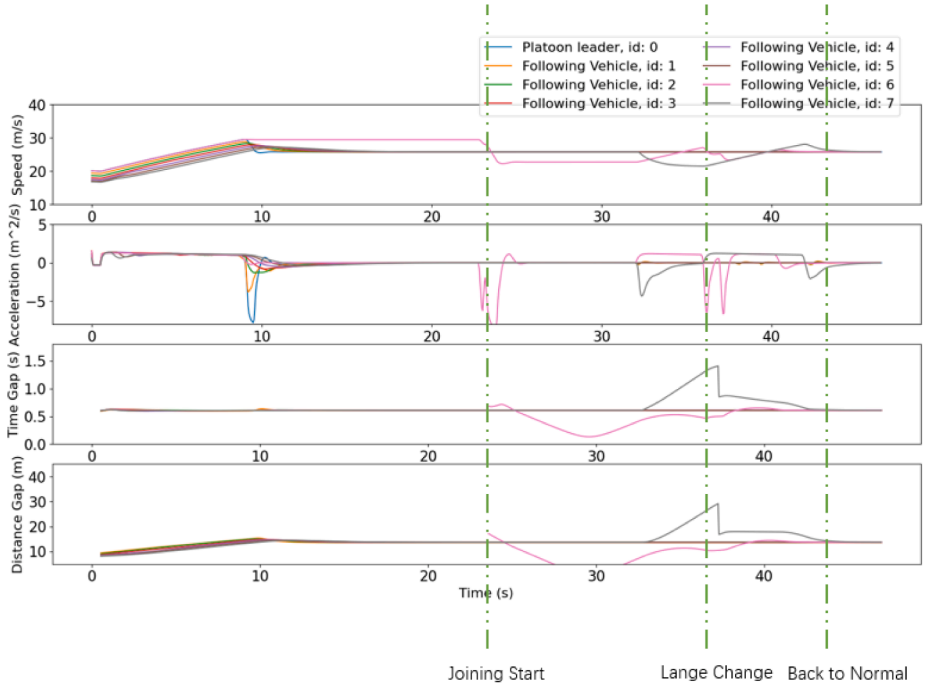
The authors confirm contribution to the paper as follows: study conception and design: Jiaqi Ma, Xu Han, Runsheng Xu, Xin Xia, Yi Guo, Pavle Bujanović, Ed Leslie; data collection: Yi Guo, Runsheng Xu, Xu Han; analysis and interpretation of results: Jiaqi Ma, Xu Han, Runsheng Xu, Xin Xia, Yi Guo, Pavle Bujanović, Ed Leslie, Mohammad Goli; draft manuscript preparation: Jiaqi Ma, Xu Han, Runsheng Xu, Xin Xia, Yi Guo, Pavle Bujanović, Ed Leslie. All authors reviewed the results and approved the final version of the manuscript.

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.



(a) Cooperative merge using HJM. ID 4 is the merging vehicle



(b) Cooperative merge using GFS. ID 6 is the merging vehicle

Fig. 19. © 2022 UCLA Mobility Lab.
Cooperative merging results utilizing different algorithms.

Data availability

Data will be made available on request.

Acknowledgments

This material is based upon work supported by the Federal Highway Administration (FHWA) under its Saxton Transportation Operations Lab (contract number DTFH6116D00030), and by the National Science Foundation through Grants CMMI No. 1901998. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of FHWA.

References

Adebisi, A., Liu, Y., Schroeder, B., Ma, J., Cesme, B., Jia, A., Morgan, A., 2020. Developing Highway Capacity Manual Capacity Adjustment Factors for Connected and Automated Traffic on Freeway Segments. Transp. Res. Rec. 2674 (10), 401–415.

Crenshaw, H.C., Edelstein-Keshet, L., 1993. Orientation by helical motion—II. Changing the direction of the axis of motion. Bull. Math. Biol. 55 (1), 213–230.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V., 2017. CARLA: An open urban driving simulator. In: Conference on Robot Learning. PMLR, pp. 1–16.

Dupuis, M, 2010. Opendrive format specification. VIRES Simulationstechnologie GmbH.

Fan, H., Zhu, F., Liu, C., Zhang, L., Zhuang, L., Li, D., Zhu, W., Hu, J., Li, H. and Kong, Q., 2018. Baidu apollo em motion planner. arXiv preprint arXiv:1807.08048.

Fellendorf, M., Vortisch, P., 2010. Microscopic traffic flow simulator VISSIM. In: Fundamentals of Traffic Simulation. Springer, New York, NY, pp. 63–93.

Firoozi, R., Zhang, X., Borrelli, F., 2021. Formation and reconfiguration of tight multi-lane platoons. Control Eng. Pract. 108, 104714.

Guo, Y., Ma, J., 2020. Leveraging existing high-occupancy vehicle lanes for mixed-autonomy traffic management with emerging connected automated vehicle applications. Transportmetrica A: Transport Science 16 (3), 1375–1399.

Hidalgo, C., Lattarulo, R., Flores, C., Pérez Rastelli, J., 2021. Platoon merging approach based on hybrid trajectory planning and CACC strategies. Sensors 21 (8), 2626.

Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y. and Azumi, T., 2018, April. Autoware on board: Enabling autonomous vehicles with embedded systems. In 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS) (pp. 287-296). IEEE.

Kesting, A., Treiber, M., Helbing, D., 2010. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 368 (1928), 4585–4605.

Liu, H., Xiao, L., Kan, X.D., Shladover, S.E., Lu, X.Y., Wang, M., Schakel, W., van Arem, B., 2018. Using Cooperative Adaptive Cruise Control (CACC) to Form High-Performance Vehicle Streams. FINAL REPORT.

Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E., 2018. Microscopic traffic simulation using sumo. In: In 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 2575–2582.

Mena-Oreja, J., Gozalvez, J., 2018. Permit-a SUMO simulator for platooning maneuvers in mixed traffic scenarios. In: In 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 3445–3450.

Milanés, V., Shladover, S.E., 2014. Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data. Transportation Research Part C: Emerging Technologies 48, 285–300.

Milanés, V., Shladover, S.E., 2015. Handling cut-in vehicles in strings of cooperative ACC Vehicles. J. Intell. Transp. Syst 20, 1–14.

On-Road Automated Driving (ORAD) committee, 2020. SAE J3216 standard: Taxonomy and definitions for terms related to cooperative driving automation for on-road motor vehicles. In SAE International.

Pizarro, G., Núñez, F., 2021. Graph-based distributed lane-change in tight multi-lane platoons. In: In 2021 Ieee Conference on Control Technology and Applications (Ccta). IEEE, pp. 1031–1036.

Porfyri, K.N., Mintsis, E., Mitsakis, E., 2018. Assessment of ACC and CACC systems using SUMO. EPiC Series in Engineering 2, 82–93.

Punzo, V., Borzacchiello, M.T., Ciuffo, B., 2011. On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data. Transportation Research Part C: Emerging Technologies 19 (6), 1243–1262.

Russell, S.J., 2010. Artificial intelligence a modern approach. Pearson Education, Inc.

Sathyan, A., Ma, J., Cohen, K., 2021. Decentralized cooperative driving automation: a reinforcement learning framework using genetic fuzzy systems. Transportmetrica B: Transport Dynamics 9 (1), 775–797.

Soleimaniamiri, S., Li, X.S., Yao, H., Ghiasi, A., Vadakpat, G., Bujanovic, P., Lochrane, T., Stark, J., Hale, D., Racha, S., 2021. Cooperative Automation Research. CARMA Proof-of-Concept Transportation System Management and Operations Use Case 4-Dynamic Lane Assignment No. FHWA-HRT-21-068.

Stevens, A., Hopkin, J., 2012. Benefits and deployment opportunities for vehicle/roadside cooperative. ITS.

Taylor, C.R., Carter, J.M., Huff, S., Nafziger, E., Rios-Torres, J., Zhang, B., Turcotte, J., 2022. Evaluating Efficiency and Security of Connected and Autonomous Vehicle Applications. In: In 2022 IEEE 19th Annual Consumer Communications & NetworkIng Conference (CCNC). IEEE, pp. 236–239.

Uno, A., Sakaguchi, T., Tsugawa, S., 1999. A merging control algorithm based on inter-vehicle communication. In: Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat, No. 99TH8383). IEEE, pp. 783–787.

Wang, Z., Wu, G., Barth, M., 2018. Distributed consensus-based cooperative highway on-ramp merging using V2X. SAE Technical Paper communications (No. 2018–01-1177.

Xu, R., Guo, Y., Han, X., Xia, X., Xiang, H., Ma, J., 2021. OpenCDA: an open cooperative driving automation framework integrated with co-simulation. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, pp. 1155–1162.