Cooperative Driving of Connected Autonomous Vehicles Using Responsibility-Sensitive Safety (RSS) Rules

Mohammad Khayatian[1], Mohammadreza Mehrabian[1], Harshith Allamsetti[2], Kai Wei[3], Po Yu[3], Chung-Wei Lin[3], Aviral Shrivastava[1]
[1]Arizona State University, [2]Western Digital, [3]National Taiwan University

ABSTRACT

Connected Autonomous Vehicles (CAVs) are expected to enable reliable and efficient transportation systems. Most cooperative driving approaches for CAVs and motion planning algorithms for multi-agent systems are not completely safe because they implicitly assume that all vehicles/agents will execute the expected plan with a small error. This assumption, however, is hard to keep since CAVs may have to slow down (e.g. to yield to a jaywalker) or are forced to stop (e.g. break down), sometimes even without a notice. Responsibility-Sensitive Safety (RSS) defines a set of safety rules for each driving scenario to ensure that a vehicle will not cause an accident irrespective of other vehicles' behavior. However, RSS rules fall short to cover many scenarios such as merges, intersections, and unstructured roads. In addition, deadlock situations can happen that are not considered by the RSS. In this paper, we propose a generic version of RSS rules for CAVs that can be applied to any driving scenario. We integrate the proposed RSS rules with the CAV's motion planning algorithm to enable cooperative driving of CAVs. Our approach can also detect and resolve deadlocks in a decentralized manner. We have conducted experiments to verify that a CAV does not cause an accident no matter when other CAVs slow down or stop. We also showcase our deadlock detection and resolution mechanism. Finally, we compare the average velocity and fuel consumption of vehicles when they drive autonomously but not connected with the case that they are connected.

KEYWORDS

Connected Autonomous Vehicles, City-Wide Traffic Management, Intelligent Transportation Systems

1 INTRODUCTION

Autonomous Vehicles (AVs) have the potential to make transportation safer by reducing the number of accidents that are caused due to human error. When AVs become connected (which are referred to as Connected Autonomous Vehicles (CAVs)), they can further improve road safety by sharing their information with each other such as position, velocity, future plans, etc. In addition, CAVs are projected to improve fuel consumption, travel time, and passenger comfort through cooperative driving.

Achieving cooperative behaviors among robots is typically studied under multi-agent motion planning in the robotics domain [1, 2]. Existing techniques can be categorized into two groups: i) centralized [3] and ii) decentralized (distributed) [4]. In centralized approaches, it is assumed that a central planner exists that has access to all information and computes the trajectory for robots e.g. path-velocity decomposition technique, while in decentralized approaches, each robot is assumed to have incomplete information and autonomously determines a plan while avoiding static and moving obstacles as well as other robots. Although centralized approaches can find the optimal solution, they are computationally demanding and less tolerant of uncertainty. On the same lines, in the Intelligent Transportation System (ITS) domain, centralized and decentralized techniques [5] are proposed where CAVs share their information with each other (through V2V) or the infrastructure (through V2I) to perform traffic management at intersections [6–9], merges [10–12].

In general, existing motion planning algorithms and traffic management techniques consider a safety buffer around each vehicle to cover for uncertainties in the localization and trajectory tracking, and then a reference trajectory is determined. A trajectory is considered to be safe if the safety buffer of the vehicle does not overlap with obstacles or other vehicles' safety buffer at any point in time. While reasonable, this definition may not provide absolute safety because it implicitly assumes that all vehicles will follow the plan (with small errors that are within the safety buffer). However, any disruption in the plan can cause an accident. For example, consider a scenario when two vehicles are driving on a street, one behind the other. If the front vehicle suddenly stops for any unplanned reason (e.g. yielding to a jaywalker), then the rear vehicle may hit the front car. In common driving parlance – the rear vehicle should not tail-gate the front vehicle.

Responsibility-Sensitive Safety (RSS) approach [13] from Mobileye+Intel addresses the safety issue from the legal/blame perspective and allows vehicles that have the right-of-the-way according to the rules of the road to change their plans. RSS proposes a set of safety rules such that if a vehicle abides by these rules, then it cannot be blamed for an accident. In the scenario that is mentioned above, RSS rules are used to determine the minimum distance at which the rear vehicle should follow the front one so that it will be able to stop without causing an accident even in the worst-case scenario. RSS uses a lane-based coordinate system to define lateral and longitudinal distances between vehicles depending on the driving scenario. For example, there is a longitudinal rule for the scenario when two vehicles are one in front of the other, and there is a lateral rule for the scenario in which two vehicles are driving in parallel to each other. The longitudinal direction is toward the center-line of the lane and the lateral direction is perpendicular to the center-line of the lane. The main shortcoming of RSS is that it is scenario-based and not all scenarios are covered because longitudinal and lateral distances cannot be computed for merges, intersections, and unstructured roads where lane markings are not provided. The first contribution of this paper is to provide a trajectory-based definition for RSS rules, that works in all situations, including merges, intersections, and unstructured roads.

When CAVs interact with each other in different scenarios, they may face a deadlock situation where CAVs yield to each other for an indefinite time. Researchers have proposed methods to detect and resolve deadlocks at intersections [8, 14, 15] and roundabouts [16]. In existing approaches, the intersection/roundabout area is divided into a grid of zones, and vehicles that intend to occupy the same zone are said to have a conflict. Then, the dependencies between CAVs (who should enter a conflict zone first and who enters second) are represented with a directed graph, and deadlocks are resolved by removing cycles in the graph. One of the limitations of existing approaches is that they use fixed zones to detect conflicts between vehicles and the size of each zone affects the efficiency and computational complexity of the conflict detection algorithm since using coarse grids makes the schedule pessimistic and using fine grids increases the number of checks. Furthermore, in existing approaches, the dependency graph is computed individually by each CAV, which is extremely inefficient because the same computing is done redundantly and the overhead grows as the number of vehicles increases. As the second contribution of this paper, we propose an efficient and decentralized approach to detect and resolve deadlock where each CAV determines only its own conflicts.

In this paper, we present a cooperative driving and deadlock resolution approach for CAVs. Instead of a lane-based coordinate system, we use future trajectories of CAVs to represent their conflicts, which can be applied to any road geometries and situations. Inspired by the RSS legal/blame perspective, we develop a new set of safety rules for CAVs to guarantee that no accidents happen if CAVs abide by proposed RSS rules. We also provide an efficient and decentralized deadlock detection and resolution algorithm for CAVs. The integration of the proposed RSS safety rules and deadlock resolution algorithms with motion planning is also provided. Results from conducting experiments on our realistic simulator –that considers vehicle dynamics and network delay– demonstrate that all CAVs remain safe even if one or more CAVs slow down or stop at any point in time. We evaluate the efficiency of our approach by comparing the average travel time of CAVs with a case that vehicles are autonomous but not connected. Finally, we showcase our deadlock resolution mechanism for an intersection scenario.

In section 2, we study related works and point out their limitation, and in section 3, we present a new version of RSS rules. Our proposed cooperative driving approach is presented in section 4. Section 5 shows the experimental results and finally in section 6, we conclude with suggestions for future works.

2 RELATED WORK

In the ITS domain, many researchers have proposed methods to cooperatively manage CAVs at intersections [6, 8, 17, 18], roundabouts [19], ramp-merging [10–12], performing cooperative lane changing [20, 21], forming platooning in highways [22, 23]. Such approaches can only be applied to a specific scenario and do not scale. There have been a number of cooperative approaches that are not scenario-based. In the method proposed by During *et al.* [24, 25], the ego CAV first determines a set of possible maneuvers that can resolve the conflict and then, select the one that has the lowest cost. The cost is determined based on energy consumption, time of maneuver, and driving comfort. In another work, Chen *et al.* [26] proposed a cooperative driving algorithm where the driving information of neighboring CAVs is obtained and the desired velocity is predicted using a Recursive Neural Network (RNN). A motion planner is developed using the predicted velocity using a fuzzy path-following controller. These approaches, however, do not consider cases where a CAV is unable to perform the desired maneuver/follow the assigned trajectory.

In the robotics domain, many researchers have focused on multi-agent motion planning algorithms problem [1, 2]. In general, cooperative motion planning algorithms can be categorized as distributed [4] and centralized [3]. In distributed approaches, each agent computes a path such that it avoids obstacles and other agents while in centralized approaches, a central planner (could be on each agent) computes the plan for all agents by exploring the whole design space. In general, distributed approaches are more popular as they require less computation and more resilient to changes in the plan or uncertainty. Existing motion planning algorithms for multi-agent systems and traffic management approaches for CAVs provide safety proofs based on the assumption that all agents stick to their plan or error is small. In the real world, CAVs may have to slow down and stop due to unforeseen reasons e.g. a CAV may break down. As a result, existing techniques are not absolutely safe for CAVs.

In 2017, researchers from Mobileye proposed a set of rules called RSS [13], which determines the minimum distance that an AV must maintain from other vehicles in order to remain safe and not being blamed for an accident. RSS rules consider the worst-case scenario for other vehicles and the ego vehicle (during the response time) to provide safety guarantees. RSS rules have been used to develop a monitoring system [27] and are implemented in the Apollo [28] open-source software.

The main issue with RSS is that it uses a lane-based coordinate system and safety rules are defined based on longitudinal (towards the lane) and lateral (perpendicular to the lane) distances, which cannot be applied to all driving scenarios such as intersections, merges or unstructured road where no road markings are present. In addition, RSS rules do not consider the interaction among other CAVs and therefore, cannot detect cases where a deadlock happens.

Researchers have proposed algorithms to detect and resolve deadlocks at intersections [8, 14], roundabouts [16] and network of intersections[15]. In such approaches, a set of pre-defined zones is used to represent the occupancy of CAVs. Next, a wait-for graph is created to represent dependency between vehicles for entering conflict zones, and deadlocks are identified by detecting cycles in the graph. However, using fixed conflict zones to detect a conflict and perform deadlock resolution is either inefficient (for coarse zones) or compute-intensive (for fine zones). Furthermore, existing approaches do not consider vehicle dynamics when resolving a deadlock and assume that a deadlock is resolved in one-shot. While in reality, it takes some time for CAVs to slow down/speed up and resolve a deadlock.

Next, we will present our approach that is developed by modifying RSS rules to support all driving scenarios and represent conflict in a more efficient way. We also present a realistic deadlock detection algorithm where each CAV computes its own dependency graph and by sharing it with other CAVs, they can detect deadlocks.

3 GENERIC FORMULATION OF RSS RULES

In this section, we introduce a trajectory-based formulation for RSS rules. The advantage of this approach is that the rules are generic and can be applied to all cases, including unstructured roads.

Given the future paths of CAVs are known, each CAV can determine the set of conflict zone C. A conflict zone, $C_i \subset C$ is defined as a convex contour that includes a subset of two CAVs' future path (FP) where the distance between the future paths is less than a threshold, d_{th}. Since two CAVs may have more than one conflict, only consecutive edges that have a distance of less than d_{th} are considered to be a part of the same conflict zone. The midpoints of the edges are used to calculate the distance between two edges from two future paths. To specify the boundaries of a conflict zone, midpoints of first and last edges are used.

Based on the road geometry and rules of the road, every pair of CAVs can determine who has the advantage to enter the conflict zone first and who has the disadvantage. For simplicity, we assume the CAV with the earlier arrival time has the advantage. Without loss of generality, we assume that one of the CAVs has the advantage and the other one has disadvantage. We represent the distance of the CAV with the advantage from the beginning of the conflict zone and from the end of the conflict zone with d^A_{begin} and d^A_{end}, respectively. Similarly, we represent the distance of the CAV with disadvantage from the beginning of the conflict zone with d^D_{begin}. Figures 1, 2, and 3 show different scenario where the d^A_{begin}, d^A_{end} and d^D_{begin} are shown. We assume that Equation1 represents the dynamics of each CAV. We assume the following vehicle dynamics for the CAV.

$$\begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{\phi} = \frac{v}{L}\tan(\psi) \\ \dot{v} = a \end{cases} \tag{1}$$

where x and y represents the position of the ego CAV in Cartesian coordinates, ϕ is the CAV's heading angle from the x-axis, v and a are linear velocity and acceleration of the CAV respectively, L is CAV's wheelbase distance and ψ is steering angle of front wheels with respect to the heading of the CAV. In order to make the model more realistic, we consider an upper bound and a lower bound on the acceleration rate and steering angle of a CAV as: $a \in [a_{min}, a_{max}]$ and $\psi \in [\psi_{min}, \psi_{max}]$ where a_{max} and a_{min} are the maximum acceleration and deceleration rates and ψ_{max} and ψ_{min} are the maximum and minimum steering angles of the vehicle.

For simplicity, the trajectory of each CAV is projected onto its path and represented with the double-integrator model. As a result, the stop distance of the CAV with advantage is calculated as:

$$d^A_{stop} = \frac{v_A^2}{2|a_{brake}|} \tag{2}$$

We assume that each CAV broadcast its information every T milliseconds and the worst-case end-to-end delay (ρ) is $2T$. Taking into account the delay, the worst-case stop distance of the CAV with disadvantage is calculated as:

$$d^D_{stop} = v_D\rho + \frac{1}{2}a_{ACC}\rho^2 + \frac{(v_D + a_{ACC}\rho)^2}{|2a_{brake}|} \tag{3}$$

The first two terms ($v_D\rho$ and $\frac{1}{2}a_{ACC}\rho^2$) indicate that the CAV with disadvantage may be accelerating in the worst-case scenario while waiting for broadcast information from the CAV with advantage. If the distance of the CAV with advantage form the end of the conflict zone is greater or equal to the stop distance of the CAV with advantage ($d^A_{end} \geq d^A_{stop}$), there is a possibility that it may slow down and stop inside the conflict zone and block the CAV with the disadvantage. Otherwise, there is no conflict. Accordingly, we define the modified RSS rule as:

Definition 1. ***General RSS Rule:*** *Given the order of entering a conflict zone is known, the minimum safe distance to maintain from the conflict zone (d^D_{SAFE}) for the CAV with disadvantage is:*

$$d^D_{SAFE} = \begin{cases} d^D_{stop} - d^A_{scenario} + \frac{VL_A + VL_D}{2} & \text{if } d^A_{end} > d^A_{stop} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $d^A_{scenario}$ is the scenario-dependent distance that the CAV with advantage travels inside the conflict zone, and VL_A and VL_D are the vehicle length for the vehicle with advantage and disadvantage, respectively. Since the distance values are calculated based on the center of CAVs, the term $\frac{VL_A+VL_D}{2}$ is added.

Lemma 3.1. *If the CAV with disadvantage always maintains a distance of at least d^D_{SAFE} from its conflict zone, it will not hit the CAV with advantage even if it changes its plan and decelerates at any point in time.*

Proof. If the distance of the CAV with the advantage from the end of the conflict zone is smaller than its stop distance, $d^A_{end} < d^A_{stop}$, it will stop outside of the conflict zone even if it decelerates at a rate of smaller than or equal to a_{brake}.

If the distance of the CAV with the advantage from the end of the conflict zone is greater than its stop distance, $d^A_{end} > d^A_{stop}$, it may stop inside the conflict zone if it decelerates. In this case, the CAV with disadvantage will be notified after ρ milliseconds in the worst-case scenario. If the CAV with disadvantage accelerates at a rate of smaller than or equal to a_{ACC} during this time interval (ρ) and then decelerates at a rate of a_{brake}, its stop distance will be equal to d^D_{stop} (Eq. (3)) and it will not enter the conflict zone and no accident will happen. For scenarios where the scenario-dependent distance is not zero, $d^A_{scenario} > 0$ (same lane and merge), the paths of the CAVs overlap and if the CAV with advantage decelerates, it will allow the CAV with disadvantage to travel through the conflict zone by $d^A_{scenario}$ and still be safe. As a result, the required safe distance is $d^D_{stop} - d^A_{scenario}$. □

Next, we study a few case studies and show how the safe RSS distance is calculated for each scenario.

3.1 Same Lane

Let us consider a scenario where two CAVs are driving in the same lane as depicted in Figure 1. The front CAV has the advantage since

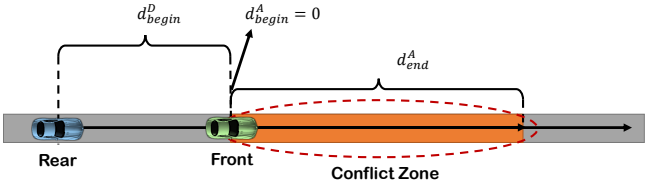

Figure 1: An example of a same lane scenario with two CAVs. The front CAV has the advantage and its distance from the conflict zone is zero. The conflict zone is highlighted in orange.

its arrival time at the conflict zone is smaller than the rear CAV. Since the paths of the front CAV overlaps with the path of rear CAV, $d^A_{scenario} = d^A_{stop}$, which means the front CAV travels d^A_{stop} meters inside the conflict zone before a complete stop and the rear CAV has d^A_{stop} meters more to stop. According to Equation (4), the required safe distance for the rear CAV (d^D_{SAFE}) to maintain from the conflict zone/front CAV is:

$$d^D_{SAFE} = d^D_{stop} - d^A_{stop} + \frac{VL_D + VL_A}{2}$$

d^D_{stop} and d^A_{stop} are calculated according to Equation2 and 3.

3.2 Intersection

Now, let us consider a scenario where two CAVs approach an intersection and their future path crosses inside the intersection area as it is depicted in Figure 2. We assume the arrival time of the green

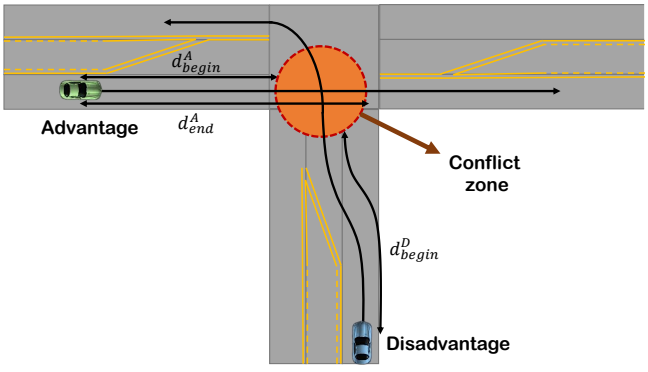

Figure 2: A scenario with two CAVs approaching an intersection and their future path intersect. It is safe to enter the conflict zone after the other CAV leaves conflict zone.

CAV to be earlier than the blue CAV and therefore, it has the advantage. If the green CAV stops anywhere inside the conflict zone, it's not safe for the blue CAV to enter the conflict zone. Therefore, the scenario-dependent distance is zero, $d^A_{scenario} = 0$. As a result, we have:

$$d^D_{SAFE} = \begin{cases} d^D_{stop} + \frac{VL_A + VL_D}{2} & \text{if } d^A_{end} \geq d^A_{stop} \\ 0 & \text{otherwise} \end{cases}$$

If the distance of the green CAV from the end of the conflict zone is smaller than its stop distance, even in the worst-case (if it decelerates at the maximum rate), it will stop outside the conflict zone and does not cause a conflict for the blue CAV. In this case, there will be no conflicts and $d^D_{SAFE} = 0$.

3.3 Merge

Next, we consider a merge scenario where two CAVs merge into the same lane as it is shown in Figure 3. Without loss of generality,

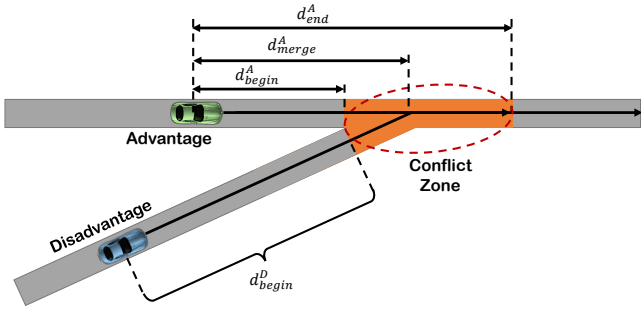

Figure 3: A scenario where two CAVs are expected to be merged into the same lane. The CAV with earlier arrival time has the advantage.

we assume one of the CAVs (green one) has the advantage and the other CAV has disadvantage respectively. In this scenario, the scenario-dependent distance is $d^A_{scenario} = \min(0, d^A_{stop} - d^A_{merge})$, where d^A_{merge} is the distance of the CAV with advantage from the merging point, which is indicated in Figure 3. As a result, the blue CAV must maintain a minimum distance of

$$d^D_{SAFE} = d^D_{stop} - \min(0, d^A_{stop} - d^A_{merge}) + \frac{VL_D + VL_A}{2}$$

from the conflict zone. Note that once the blue CAV reaches the merge point, the $d^A_{scenario}$ is changed. The lateral case in the original RSS rules (two CAVs driving on adjacent lanes) can be modeled like a merging case. If any of the CAVs attempts to merge into the other CAV's lane, it is only allowed if the created conflict zone is far enough from the other CAV i.e. at least d_{max}.

4 PROPOSED COOPERATIVE DRIVING APPROACH

In this section, we first present the algorithm that runs on each CAV assuming no deadlock situation happens. In the next section, we explain the deadlock resolution algorithm.

4.1 Main Algorithm

Given the initial position and final destination of a CAV are known, the motionPlanner uses the world's map to determine the shortest path (R) that connects CAV's current position to the destination.

We assume that at least one feasible path exists that connects CAV's current location to its destination. The map, $M(N, E)$, is a directed graph where N is the set of nodes or waypoints and E is the set of edges or connections between waypoints. Each edge of the map graph has a weight, w, which indicates the maximum velocity for that segment of the road. In our algorithm, we assume that the ego CAV's computation time and communication time are bounded by T.

In a periodic manner, each CAV broadcasts its ID, position, velocity, timestamp, and its future path (FP), which is an array of x-y coordinates. We assume that all CAVs synchronize their clock using GPS so that timestamps are captured with clocks that have almost the same notion of time. When the CAV receives the information of other CAVs, it checks if their paths intersect or the distance between their paths is less than a threshold. If so, the CAV computes a set of conflict zones (C). For each conflict zone, the CAV determines which vehicle has the advantage to enter the conflict zone first based on who is expected to reach the conflicting zone first. To detect possible deadlocks, the CAV computes a graph called Partial Dependency Graph (PDG), which represents the dependency among other CAVs and itself (who should yield to who over a conflict zone). Next, the CAV broadcasts the computed PDG, and after receiving other CAVs PDG, it constructs the Complete Dependency Graph (CDG) to detect and resolve possible deadlocks. Finally, if the CAV has disadvantage over a conflict zone, it computes a safe velocity so that it always maintains a safe distance from that conflict zone. Based on the determined velocity, the weight of some of the edges are updated to reflect the presence of other CAVs and to make sure a safe distance is always maintained from the conflict zone. Then, the motion planner runs the shortest path algorithm again to check if a shorter path exists that does not cause a new conflict. Finally, the motion controller uses a subset of future waypoints and velocities of corresponding edges to determine the desired velocity and control inputs (steering angle and acceleration) for the CAV. Alg. 1 shows the pseudo-code of our algorithm that is executed on each CAV. To have a better understanding of our algorithm, we have depicted different components of our approach and their relationship in Figure 4. Next, we will focus on explaining the functionality of each component of the algorithm.

Algorithm 1: CAVs algorithm

```
while has not reached the destination do
    FP = compute_future_path();
    CAV_info = [x, y, v, ts, FP, ID];
    broadcast(CAV_info);
    others_info = receive_other_CAVs_info();
    for each member of other_CAVs_info do
        [C, PDG] = find_conflict_zones(CAV_info,
          others_info);
    end
    broadcast(PDG);
    others_PDG = receive_other_PDGs();
    CDG = construct_CDG(PDG, other_PDGs);
    C = deadlock_resolution(C, CDG);
    if ego CAV has disadvantage over a conflict zone then
        [FP, velocity] = motion_planner(C, Map);
    end
    motionController(FP, Velocity);
end
```

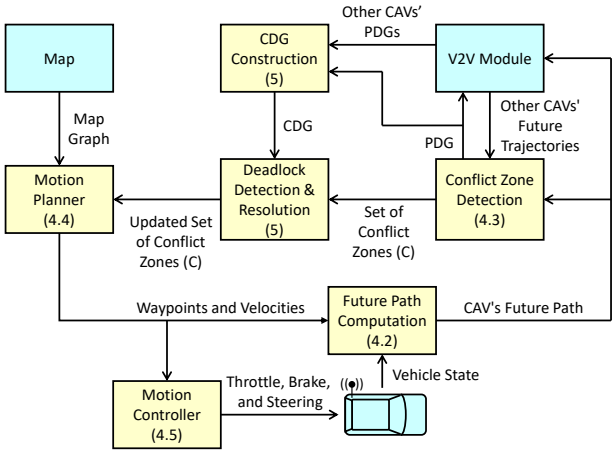

Figure 4: Overview of our approach. Details of each component –except V2X module and map– are explained later.

4.2 Future Path Computation

Each CAV broadcasts its ID, position (x, y), velocity (v), and the corresponding timestamp (ts) as well as its future path $((x_1, y_1), ..., (x_n, y_n))$. Assuming the CAV's motion controller is tuned to have a short settling time, the CAV will track its path with a negligible error. As a result, we represent the future position of the CAV with a subset of its expected route (R). Given $R \subset M(N, E)$ is the route of the CAV, the future path of the CAV, $FP \subset R$ is calculated as follows which consists from n points:

$$FP = \left\{ (x_i, y_i) \in R \,\middle|\, \left(\sum_{i=2}^{i=n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \right) < d_{max} \right\} \tag{5}$$

where d_{max} is the fixed length of the future path calculated as:

$$d_{max} = v_{max}(\rho + t_b) \tag{6}$$

ρ represents the worst-case end-to-end delay from one CAV capturing its information and broadcasting it, to another CAV's actuation based on the received information (see Figure 5) and t_b is the worst-case brake time which can be calculated as $t_b = \frac{v_{max}}{|a_{brake}|}$. Figure 5 shows the execution profile of our algorithm on two CAVs (i and j). Let us assume that CAVs i and j have a conflict and CAV i (top) has the advantage. If CAV i slows down due to any reason right after sensing and broadcasting its info, the CAV j will not be notified until receiving the next broadcast. As a result, the worst-case end-to-end delay (ρ) is bounded by $2T$ as depicted in Figure 5. By computing the d_{max} based on the worst-case info sharing delay and worst-case braking time, we ensure that for the first time that

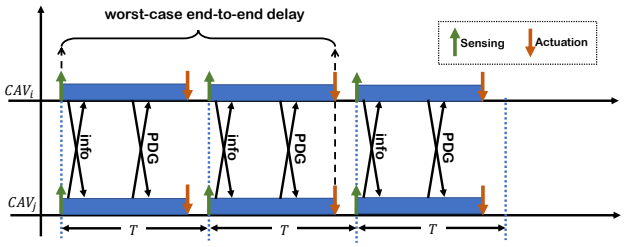

Figure 5: CAVs perform computation and communication in a synchronized manner. The worst-case sensing to actuation delay corresponds to the case that CAV_i breaks down right after sensing.

two CAVs detect that they have a conflict, the CAV with the disadvantage have enough distance to safely stop without entering the conflict zone, even in the worst-case scenario.

4.3 Conflict Zone Detection

Despite existing approaches that use fixed conflict zones, we use CAV's expected trajectory to detect a conflict zone. As mentioned before, CAVs' future paths (FP) are used to represents their expected future position. First, CAVs computes the distance between the mid point of edges on their path. All contiguous edges that have a distance less than d_{th} are considered to be a part of the same conflict zone. Two CAVs may have multiple conflicts on their path as depicted in Fig. 6. Each conflict zone C_i is a data structure that

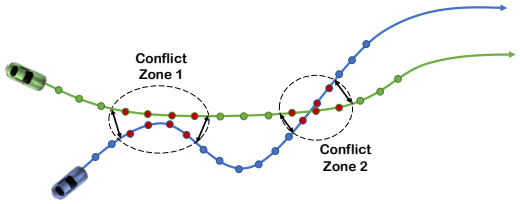

Figure 6: An example of two CAVs with arbitrary paths and two conflict zones. The conflict zone includes parts of the CAV path (waypoints) where the distance between paths of CAVs is less than a threshold.

includes waypoints that are inside the conflict zones, distance of CAVs from the the beginning and end of the conflict zone, their expected arrival time at the conflict zone (Equation 7) and the ID of the CAV that has the advantage. We compute the arrival time assuming the CAV drives at a constant velocity.

$$TOA_i = \frac{d^i_{begin}}{v_i} \tag{7}$$

where d^i_{begin} is the distance of the CAV i from the conflict zone and v_i is the velocity of the CAV i. Since the algorithm is executed periodically (every T ms), the value of TOA_i is updated as the velocity of the CAV changes. If a CAV is stopped inside a conflict zone, its arrival time is set to zero. By default, the CAV with the earliest arrival time will have the advantage unless it is changed to resolve a deadlock (explained in the next section) or the other CAV has a priority (e.g. opposite direction). If two CAVs have the same arrival time, the CAV with the lower ID will have the advantage to break the tie. In addition, if the difference between the arrival times of two CAVs is within the accuracy of the clock synchronization (± 10 nanoseconds for GPS), they use CAVs ID to determine who has the advantage.

4.4 Motion Planner

If a CAV has disadvantage over a conflict zone, it first checks if an alternative path exists such that it avoids all the conflicts. If such a path exists, the CAV selects that path and if not, the CAV calculates a safe velocity (v_{SAFE}) to be maintained so that the CAV is always safe. The safe velocity, v_{SAFE}, is determined based on the minimum safe distance that the ego CAV must maintain from the conflict zone given that other CAV –which has the advantage– may slow down at any point in time and stop inside the conflict zone.

Maximum Safe Velocity: For each segment of the road that has a distance of d_C from the conflict zone, the maximum safe velocity is computed using Equation 8.

$$v_{SAFE} = \frac{-(2\rho a_{ACC} + 2|a_{brake}|) + \sqrt{\Delta}}{2} \tag{8}$$

where $\Delta = 4(a^2_{brake} + 2a_{ACC}\rho a_{brake} - a_{ACC}\rho^2 a_{brake} - 2d_C|a_{brake}|$. Equation 8 is determined by solving Equation 3 for v_D when the distance from the conflict zone is d_C. d_C can be calculated using Equation 11. Equation (8) ensures that the CAV with disadvantage has always a minimum distance of d^D_{SAFE} from the conflict zone.

Once the safe velocity is determined for each conflict zone (C_i), the motion planner updates weights of the map $M(N, E)$, to account for the presence of other CAVs and generates safe velocities for the motion controller. To account for the presence of other CAVs, the motion planner determines, U the set of all edges (e_i) that are connected to waypoints that are on the future path of other CAVs

$$U = \{e_i \in E | e_i \in connected(FP)\} \tag{9}$$

where $connected(FP)$ is the set of all edges that are connected to waypoints in the set FP. To account for the safe RSS distance, the motion planner determines U_D, the set of all edges that are connected to the waypoints that are on the future path of the CAV with disadvantage (FP_D) and are either a member of the conflict zone set (C) or within the safe distance (d^D_{SAFE}) of the conflict zone.

$$U_D = \{e_i \in E | e_i \in connected(FP^C_D)\} \tag{10}$$

where $connected(FP^C_D)$ is the set of all edges that are connected to waypoints in the set FP^C_D. Figure 7 shows a merge scenario and CAV's future paths. Weights of all edges connected to nodes that are on the path of the CAV with advantage (depicted in green) and all edges that are on the path of the ego CAV and are either within the safe distance of or inside the conflict zone are updated. The set U_D and U are highlighted on the path of CAVs. The subset of future point, FP^C_D, is determined as:

$$FP^C_D = \{n_i \in N | n_i \in FP_D \; and \; n_i \in C \; or \; n_i \in within(C_j)\}$$

$within(C_j)$ is the set of all waypoints that where their distance from the conflict zone j is less than d_{SAFE}. To calculate the distance

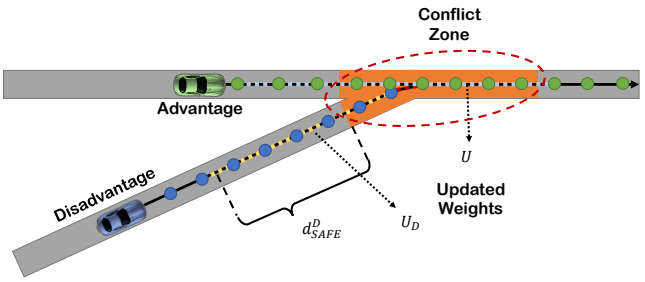

Figure 7: Weights of the edges on the path of the other CAV and edges on path of the ego CAV are updated to account for the presence of other CAVs as well as the conflict zone and the required safe distance.

between two waypoints, we use the following equation:

$$distance = \sum_{i=2}^{N} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{11}$$

where N is the number of waypoints including the first and last waypoints. Finally, weights of each edge in set U and U_D are updated based on their distance from the conflict zone using Equation 8:

$$w_i = \frac{l}{v^i_{SAFE}} \tag{12}$$

where i refers to each segment of the road, l is the length of the corresponding edge and v^i_{SAFE} is the safe velocity calculated for each segment of the road (edge). Since the weight of an edge may be updated multiple times –as it may be involved in more than one conflict–, the maximum weight is considered (the slowest safe velocity) for an edge. If the safe velocity (v_{SAFE} is equal to zero, instead of infinity, the weight is set to be a large constant number.

After updating the weights, the motion planner uses the Dijkstra algorithm to find the shortest path to the destination. The summation of weights ($\sum w_i$) from the source to the destination corresponds to the travel duration.

4.5 Motion Controller

The motion controller uses the future waypoints and safe velocities to calculate the reference heading angle θ_{ref} and the safe velocity v_{ref} for the CAV. For the desired heading angle (θ_{ref}), the motion controller selects a look-ahead point similar to the pure pursuit algorithm [29] and calculates the bearing angle from its current location (x, y) to the look-ahead point:

$$\theta_{ref} = atan2(x - x_l, y - y_l) \tag{13}$$

where x_l and y_l correspond to the x-y coordinate of the look-ahead point. We assume that each vehicle has an initial desired velocity of v_0 and never drives faster than that. The motion controller uses the weight of the next edge to determine the reference velocity ($v_{ref} = \frac{d_i}{w_i}$). If the calculated velocity is greater than CAV's initial desired velocity (v_0), it sets the reference velocity to be v_0. If the reference velocity is close to zero, ($v < \epsilon$), it is set to zero. Once the reference heading and velocity are calculated, they are passed to two Proportional Integral Derivative (PID) controllers to calculate the steering angle and acceleration for the vehicle:

$$\begin{cases} \psi = k_P e_\theta + k_I \int e_\theta + k_D \dot{e}_\theta \\ a = k'_P e_v + k'_I \int e_v + k'_D \dot{e}_v \end{cases} \tag{14}$$

where k_P, k_I, k_D and k'_P, k'_I, k'_D are constant (controller gains) that are tuned to achieve a fast response while the overshoot is small (short settling time), $e_\theta = \theta_r - \theta$ and $e_v = v_{SAFE} - v$, and $\dot{e}_v$ and $\dot{e}_\theta$ are the derivative of e_v and e_θ, respectively.

5 DEADLOCK DETECTION AND RESOLUTION

In order to detect and resolve deadlocks, all CAVs create a directed graph called the *dependency graph*. Nodes of the dependency graph are vehicle IDs and edges indicate that if a CAV is yielding to another CAV over a conflict zone. There will be a directed edge from node V_i to node V_j if CAV V_i is yielding to the CAV V_j over a conflict zone. Since a CAV determines only the conflicts between itself and other CAVs –and not the conflicts between other CAVs, the constructed dependency graph is not complete. We refer to the dependency graph of each CAV as the "partial dependency graph" or PDG. To compute the complete graph, each CAV broadcasts its PDG to inform other CAVs about its conflict zones with other CAVs and to receive other CAVs' PDG. From the received PDGs of other CAVs and the PDG of the ego CAV, the complete dependency graph (CDG) is constructed. To build the CDG, the PDG is incrementally updated by adding nodes and edges for each received PDG. Finally, all edges between two nodes are merged into one. Figure 8 shows a scenario with 5 CAVs that have determined their PDG and the final consensual CDG.

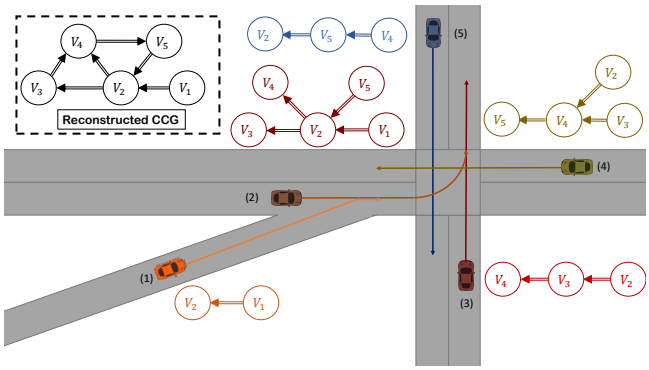

Figure 8: Each CAV determines and broadcasts its PDG. After receiving other CAVs' PDG, CAVs construct the CDG and can resolve deadlocks.

After constructing the CDG, each CAV checks if the CDG has a cycle. We use the Depth-First Search (DFS) algorithm to detect cycles. If a deadlock is detected, each CAV calculates a score for each CAV that is involved in the cycle based on its average time of arrival at corresponding conflict zones. If a CAV has m conflicts, its score is calculated as:

$$S = \frac{\sum_{i=1}^{m} TOA_i}{m}$$

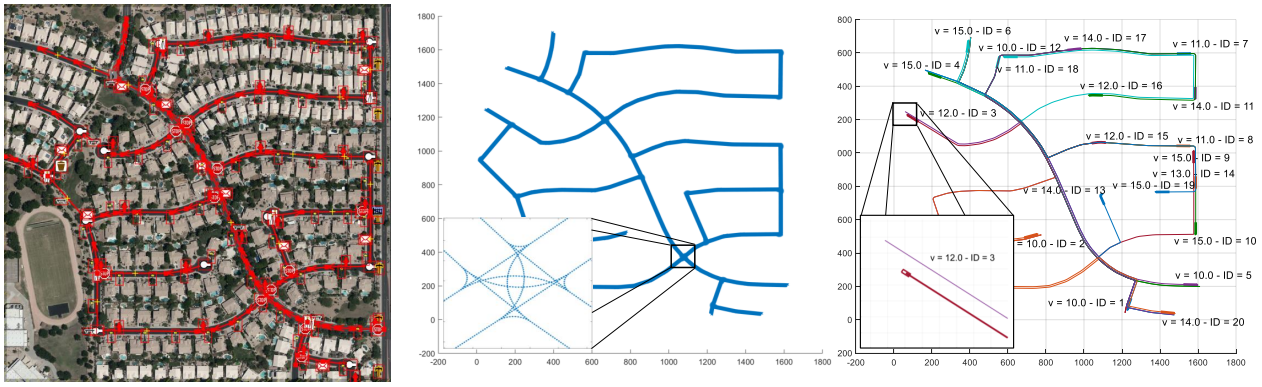

Figure 9: A snapshot of a map retrieved from the OpenStreetMap (left), its corresponding directed graph in MATLAB (middle) and a scenario with randomly spawned vehicles on the map (Right).

where TOA_i is the time of arrival of the CAV at its ith conflict zone. We select the CAV with the least average time of arrival to have the advantage over all of its conflict zones because on average, it can reach its conflict zone earlier than others. We refer to this CAV as the leader. Once the leader is determined, the direction of all incoming edges to the leader's node is reversed. If two CAVs have the same score, the CAV with the lower ID number will be selected as the leader. Since there can be more than one cycle in a graph, this process is repeated until all cycles are removed.

Lemma 5.1. *If the CDG has no cycles, then there is no deadlock involving the ego CAV.*

Proof. Once the CDG is modified to be acyclic, there is no path (set of sequential edges) starting at node V_{ego} that eventually loops back to node V_{ego} again, which means the ego CAV never yields to other CAVs that are yielding to the ego CAV and therefore, there is no deadlock involving the ego CAV. □

It takes some time to resolve a deadlock due to the vehicle's dynamic –CAVs cannot change their velocity and expected arrival time instantly. As a result, CAVs may face the same deadline again when they compute the CDG after T. It can be shown that the result of deadlock resolution will be the same (the same CAV will be selected as the leader) until the deadlock is resolved. Since the leader has the least average time of arrival in the first iteration, it does not yield to any other CAV while other CAVs involved in the deadlock slow down to yield to at least one CAV. Therefore, the average time of arrival of the leader will be less than other CAVs in the second iteration and so on.

6 EXPERIMENTAL RESULT

We evaluated our algorithm on a simulator that is developed in Matlab. We created a tool in Python to automatically extract a desired map from the OpenStreetMap[1] (OSM format) and then generate the world map graph for it. Once the map is generated, a driving scenario is created where initial position and velocity and the destination of CAVs are randomly selected. We used differential equations represented in (1) to model vehicle's dynamics. The size of each vehicle is 5x2 m, the lane width is 5 m and the distance between waypoints of the map is 0.5 m. Gains of the controller for both heading and velocity control are $K_P = 5$ and $K_D = 0.1$. Other parameters of the vehicle are listed in Table 1. CAVs communication delay is modeled by queuing the broadcast packets. In Figure 9, a randomly generated map from openStreetMap, its corresponding map graph and a random scenario with 20 CAVs are depicted.

v_{min}	v_{max}	a_{min}	a_{max}	ψ_{min}	ψ_{max}	T	ρ
$0\frac{m}{s}$	$23\frac{m}{s}$	$-8\frac{m}{s^2}$	$5\frac{m}{s^2}$	$-\frac{\pi}{3}$ rad	$\frac{\pi}{3}$ rad	$0.1s$	$0.2s$

Table 1: Parameters of the CAVs for simulation.

6.1 Safety Evaluation

To demonstrate the safety of the proposed algorithm, we created a merge and an intersection scenario where two CAVs have a conflict on their future path as it is depicted in Figure 10.

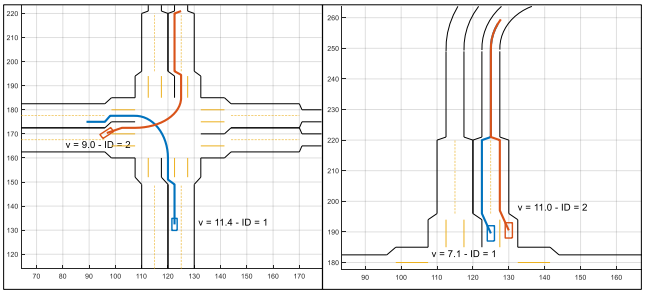

Figure 10: An Intersection and a merge scenario are created. The CAV with advantage suddenly decelerates and stops.

To verify that CAVs are always safe, we force the CAV with the advantage to suddenly decelerate at different times. We show that no accident will happen regardless of the deceleration time and CAVs maintain a minimum safe distance of 5 meters. Using brute-force testing, the deceleration time of the CAV with the advantage varies in a 30-second interval with a 0.1 s increment that

[1] https://www.openstreetmap.org/

includes critical times that stops inside the conflict zone. Figure 11 and 12 show the distance between CAVs for the intersection and merge scenarios, respectively. In the intersection scenario, the CAV

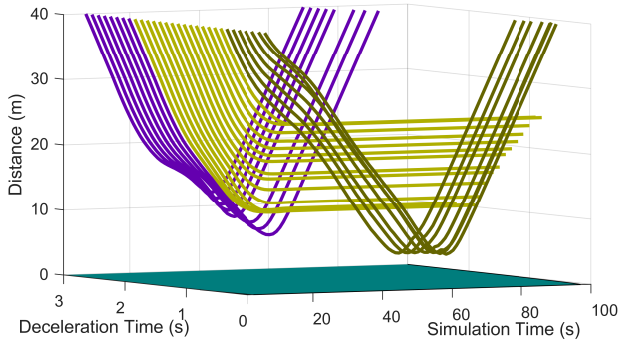

Figure 11: Brute-force evaluation of an intersection scenario. CAVs distance is always greater than a threshold regardless of the deceleration time of the CAV with the advantage –if it stops before entering the conflict zone (dark green), inside the conflict zone (yellow), or after the conflict zone (blue).

with the advantage may stop before, inside, or after the conflict zone where distances between CAVs are depicted in dark green, yellow, and blue colors, respectively. For cases that the CAV with advantage stops before or after the conflict zone, the CAV with disadvantage continues while in cases that the CAV with advantage stops inside the conflict zone, the CAV with advantage slows down and stops (depicted in yellow). In the merge scenario, the conflict

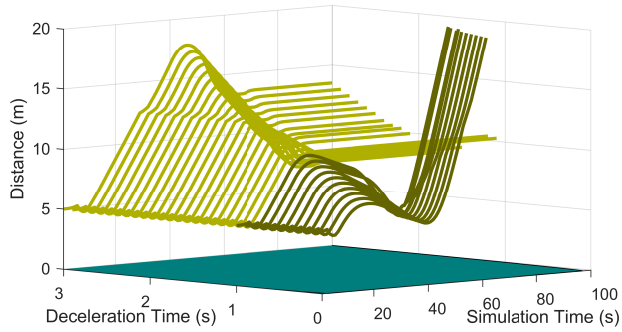

Figure 12: Merge scenario - CAVs distance remains greater than a threshold for all cases where the CAV with advantage stops before entering the merge (Yellow) or after the merge (dark green).

zone moves with the CAV with the advantage after it reaches the merging point. As a result, the CAV with the advantage either stops before the conflict zone or inside it. For cases that the CAV with the advantage stops before the merging point, the CAV with the advantage continues and enters the merge (depicted in dark green) and for the rest of the cases, the CAV with the disadvantage slows down and stops (depicted in yellow).

6.2 Deadlock Resolution Demonstration

To evaluate our deadlock detection and resolution approach, we created a deadlock situation at the intersection (Figure 13). The right part of Figure 13 shows the CCG for the scenario. We fixed the

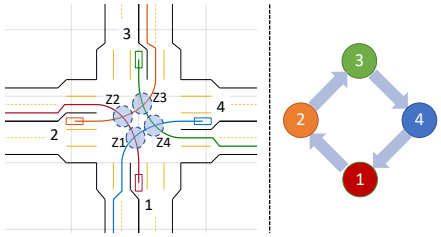

Figure 13: A deadlock scenario where 4 CAVs approach the intersection with same velocity (left) and the corresponding CDG (right).

paths of CAVs to make a left turn at the intersection while having the same distance from the intersection and the same velocity. We simulated CAVs' behavior with and with our deadlock detection. Figure 14 shows the velocities of CAVs for both cases. For the case that no deadlock resolution is done, CAVs slow down to yield to other CAVs and eventually stop and will wait forever. For the case with deadlock resolution, CAVs slow down at first but speed up when their conflict zone is cleared. We can observe that after 7s,

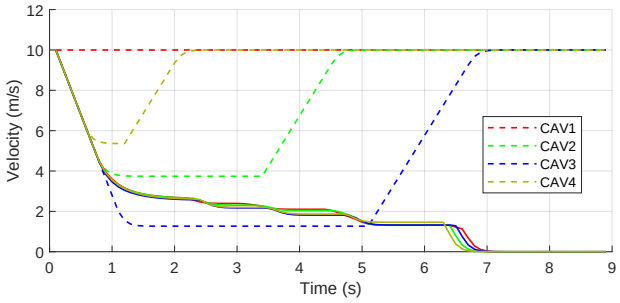

Figure 14: Velocity profiles of CAVs with and without deadlock resolution for the scenario in Figure 13

all CAVs reach their desired velocity (10m/s) while in no deadlock detection case, their velocity converges to zero.

6.3 Efficiency Evaluation

To evaluate the efficiency of our approach, we compared the performance of our approach with the case that vehicles are autonomous but not connected. For the non-connected case, the intersections are managed by stop signs and all other conflicts among CAVs are handled by the AV's perception system e.g. adaptive cruise control (ACC) system. We extracted a map from the OpenStreetMap (Figure 9) and simulated three scenarios, i) light traffic with 5 vehicles, ii) moderate traffic with 10 vehicles, and iii) heavy traffic with 20 vehicles being present at the same time. When a vehicle exits the map boundary, a new vehicle is spawned. We measured the average

velocities of CAVs and reported them in Table 2. We also computed the fuel consumption of CAVs using the following model [30] and reported them in Table 2:

$$f = \begin{cases} 0 & \text{if } P_T > 0 \\ \frac{f_i}{3600} + \beta_1 P_T + \beta_2 a P_I & \text{otherwise} \end{cases} \quad (15)$$

where $P_T = \min(P_{max}, P_C + P_I)$ is the total tractive power (kW), $P_C = b_1 v + b_2 v^3$ is the cruise component of total power (kW), $P_I = \frac{mav}{1000}$ is the inertia component of the total power (kW), $f_i = 888.8mL/h$ is the instantaneous fuel consumption rate (mL/s), P_{max} is the maximum engine power (kW), m is the vehicle mass, a and v are the instantaneous acceleration and velocity, b_1 is rolling resistant factor (kN), and b_2 is the aerodynamic drag factor (kN/(m/s)^2), β_1 and β_2 are the efficiency factors for non-accelerating and accelerating cases.

	Light Traffic		Moderate Traffic		Heavy Traffic	
	AVs	CAVs	AVs	CAVs	AVs	CAVs
Average Velocity (m/s)	10.51	11.55	10.91	11.83	11.21	11.96
Average Fuel Consumption (mL/s)	1.271	0.495	1.089	0.479	1.017	0.485

Table 2: Comparing the average velocity and fuel consumption of vehicles when they navigate autonomously (non-connected) and cooperatively (connected).

With the help of shared information, CAVs not only drive at higher velocities, they drive smoother than non-connected case because they slow down and stop less frequently and therefore, their fuel consumption is less than the connected case.

7 CONCLUSION AND FUTURE WORKS

In this paper, a new definition is introduced for the RSS rules that can be applied to any scenario, and CAVs' safety is ensured by considering the worst-case scenario. Next, we presented a cooperative driving algorithm for CAVs based on proposed RSS rules. Our algorithm can also detect and resolve deadlocks in a distributed manner. The correctness of our approach is verified by conducting experiments using our simulator. Future works include taking into account various fault models (e.g. the network delay is larger than T or a CAV is unable to communicate, a CAV is being compromised and lies about its position and/or its future trajectory, etc.) with the help of infrastructure (e.g. installed cameras).

REFERENCES

[1] MG Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, 2018.

[2] Federico Rossi, Saptarshi Bandyopadhyay, Michael Wolf, and Marco Pavone. Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*, 51(12):112–117, 2018.

[3] Jufeng Peng and Srinivas Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research*, 24(4):295–310, 2005.

[4] Kostas E Bekris, Konstantinos I Tsianos, and Lydia E Kavraki. A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3784–3790. IEEE, 2007.

[5] Mohammad Khayatian, Mohammadreza Mehrabian, Edward Andert, Rachel Dedinsky, Sarthake Choudhary, Yingyan Lou, and Aviral Shirvastava. A survey on intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(4):1–27, 2020.

[6] Bowen Zheng, Chung-Wei Lin, Shinichi Shiraishi, and Qi Zhu. Design and analysis of delay-tolerant intelligent intersection management. *ACM Transactions on Cyber-Physical Systems*, 4(1):1–27, 2019.

[7] Mohammad Khayatian, Rachel Dedinsky, Sarthake Choudhary, Mohammadreza Mehrabian, and Aviral Shrivastava. R 2 im–robust and resilient intersection management of connected autonomous vehicles. 2020.

[8] Yi-Ting Lin et al. Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–21, 2019.

[9] Mohammad Khayatian et al. Crossroads+ a time-aware approach for intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(2):1–28, 2019.

[10] Xiao-Yun Lu and J Karl Hedrick. Longitudinal control algorithm for automated vehicle merging. *International Journal of Control*, 76(2):193–202, 2003.

[11] Jackeline Rios-Torres and Andreas A Malikopoulos. Automated and cooperative vehicle merging at highway on-ramps. *Transactions on Intelligent Transportation Systems*, 18(4):780–789, 2016.

[12] Shunsuke Aoki and Ragunathan Rajkumar. A merging protocol for self-driving vehicles. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, pages 219–228. IEEE, 2017.

[13] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[14] Changliu Liu, Chung-Wei Lin, Shinichi Shiraishi, and Masayoshi Tomizuka. Distributed conflict resolution for connected autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 3(1):18–29, 2017.

[15] Florent Perronnet et al. Deadlock prevention of self-driving vehicles in a network of intersections. *IEEE Transactions on Intelligent Transportation Systems*, 20(11):4219–4233, 2019.

[16] Reza Azimi, Gaurav Bhatia, Ragunathan Raj Rajkumar, and Priyantha Mudalige. Stip: Spatio-temporal intersection protocols for autonomous vehicles. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*, pages 1–12. IEEE Computer Society, 2014.

[17] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[18] Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava. Rim: Robust intersection management for connected autonomous vehicles. In *Real-Time Systems Symposium*, pages 35–44. IEEE, 2018.

[19] Lejla Banjanovic-Mehmedovic et al. Autonomous vehicle-to-vehicle (v2v) decision making in roundabout using game theory. *Int. J. Adv. Comput. Sci. Appl*, 7:292–298, 2016.

[20] Bai Li et al. Cooperative lane change motion planning of connected and automated vehicles: A stepwise computational framework. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 334–338. IEEE, 2018.

[21] Jianqiang Nie, Jian Zhang, Wanting Ding, Xia Wan, Xiaoxuan Chen, and Bin Ran. Decentralized cooperative lane-changing decision-making for connected autonomous vehicles. *IEEE Access*, 4:9413–9420, 2016.

[22] Pedro Fernandes and Urbano Nunes. Platooning of autonomous vehicles with intervehicle communications in sumo traffic simulator. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1313–1318. IEEE, 2010.

[23] Siyuan Gong, Anye Zhou, and Srinivas Peeta. Cooperative adaptive cruise control for a platoon of connected and autonomous vehicles considering dynamic information flow topology. *Transportation Research Record*, 2673(10):185–198, 2019.

[24] Michael Duering and Patrick Pascheka. Cooperative decentralized decision making for conflict resolution among autonomous agents. In *International Symposium on Innovations in Intelligent Systems and Applications*, pages 154–161. IEEE, 2014.

[25] Michael During and Karsten Lemmer. Cooperative maneuver planning for cooperative driving. *IEEE Intelligent Transportation Systems Magazine*, 8(3):8–22, 2016.

[26] Yimin Chen, Chao Lu, and Wenbo Chu. A cooperative driving strategy based on velocity prediction for connected vehicles with robust path-following control. *IEEE Internet of Things Journal*, 2020.

[27] Mohammad Hekmatnejad et al. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *17th ACM-IEEE Conference on Formal Methods and Models for System Design*, pages 1–11, 2019.

[28] Apollo Auto. Apollo, An open Autonomous Driving Platform. https://github.com/ApolloAuto/apollo, 2019. [Online; accessed 14-Oct-2019].

[29] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[30] Rahmi Akçelik, Robin Smit, and Mark Besley. Calibrating fuel consumption and emission models for modern vehicles. In *IPENZ transportation group conference, Rotorua, New Zealand*, 2012.