Game-Theoretic Modeling of Driver and Vehicle Interactions for Verification and Validation of Autonomous Vehicle Control Systems

Nan Li, Dave Oyler, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck Girard

***Abstract*—Autonomous driving has been the subject of increased interest in recent years both in industry and in academia. Serious efforts are being pursued to address legal, technical and logistical problems and make autonomous cars a viable option for everyday transportation. One significant challenge is the time and effort required for the verification and validation of the decision and control algorithms employed in these vehicles to ensure a safe and comfortable driving experience. Hundreds of thousands of miles of driving tests are required to achieve a well calibrated control system that is capable of operating an autonomous vehicle in an uncertain traffic environment where multiple interactions between vehicles and drivers simultaneously occur. Traffic simulators where these interactions can be modeled and represented with reasonable fidelity can help decrease the time and effort necessary for the development of the autonomous driving control algorithms by providing a venue where acceptable initial control calibrations can be achieved quickly and safely before actual road tests. In this paper, we present a game-theoretic traffic model that can be used to 1) test and compare various autonomous vehicle decision and control systems and 2) calibrate the parameters of an existing control system. We demonstrate two example case studies, where, in the first case, we test and quantitatively compare two autonomous vehicle control systems in terms of their safety and performance, and, in the second case, we optimize the parameters of an autonomous vehicle control system, utilizing the proposed traffic model and simulation environment.**

***Index Terms*—Game Theory, Reinforcement Learning, Traffic Modeling, Autonomous Vehicles, Verification and Validation**

I. Introduction

ONE of the most significant challenges that must be addressed before autonomous cars can be deployed in mass production is the Verification and Validation (V&V) of their control systems in terms of safety and performance [1], [2]. It has been estimated that autonomous vehicles need to be driven 275 million miles without fatality to assure the same rate of reliability as existing human driven cars [3]. Hence testing and calibration of decision and control systems of autonomous vehicles in simulation becomes necessary to complement the on-the-road tests and the formal methods-based and reachability analysis-based procedures (e.g., see [4], [5], [6], [7]).

One common approach to the design of control systems for autonomous vehicles is to utilize a hierarchical control structure, wherein a higher level outer loop controller generates reference trajectories for the lower level inner loop controller which, in turn, determines the steering angle and acceleration/deceleration inputs required to follow the reference trajectory [8]. Designing these control systems is a challenging task as they need to provide a safe ride together with an acceptable performance and comfort in an uncertain traffic environment. Uncertainties generally emanate from unpredictable behaviors of other drivers, pedestrians, unexpected obstacles, and changing road and weather conditions. Several control approaches have been proposed for autonomous vehicles including decision trees [9], [10], Partially Observable Markov Decision Processes (POMDPs) [11] and methods based on multi-policy decision making [12], that are mainly employed as outer loop controllers. For the inner loop, one of the most common approaches is based on Model Predictive Control (MPC) [13]-[14].

Note that advanced driver behavioral models may also be used in the outer loop [15], [16], with the motivation that an autonomous vehicle should be able to drive at least as well as a human driver. In fact, some experts have suggested that autonomous vehicles should be permitted on public roads only after it is proven that they are superior to human drivers [2].

Simulators can facilitate the development and testing of autonomous vehicle control algorithms, complementing the road tests. Since autonomous vehicles will be interacting with human driven vehicles in traffic, such simulators need to reflect driver and vehicle interactions. Several methods have been proposed in the literature to address this problem. In [17] and [18], a Hidden Markov Model (HMM) based driver model is proposed based on real driving data. In [19] and [20], k-means clustering is used to determine the driving mode and define an approach to predicting and overbounding future vehicle trajectories. It is shown that the performance of an assisted driving algorithm can be improved through the prediction of driver inputs. In [21], a “cognitive architectures” approach, which is “a computational framework that incorporates built-in, well-tested parameters and constraints on cognitive and perceptual-motor processes,” is utilized for driver modeling. Built-in logical rules (if-then-else) are used to represent the human decision making process. In [22], lane change be-

N. Li, D. Oyler, M. Zhang, I. Kolmanovsky, and A. Girard are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109 USA.

Y. Yildiz is with the Department of Mechanical Engineering, Bilkent University, 06800 Cankaya, Ankara, Turkey.

Ilya Kolmanovsky and Nan Li acknowledge the support for this research of the National Science Foundation under Award Number CNS 1544844 to the University of Michigan.

Anouck Girard acknowledges the support for this research of the Air Force Research Laboratory grant FA 8650-07-2-3744 to the University of Michigan.

havior of drivers is modeled using a multi agent simulation system called "Simulation of Intelligent TRAnsport Systems (SITRAS)." Several logical algorithms are used to model the decision making during lane changes. The resulting actions of the drivers are therefore predefined with strict rules even though driver aggressiveness can be incorporated into the model by tuning certain parameters.

With respect to the existing approaches, the present paper is distinguished by the advanced modeling of driver and vehicle interactions in traffic using a specific game theoretic formulation that is scalable to multiple vehicles. The proposed method has the following advantages: a) actions of drivers and vehicles are determined by utilizing a decision making process, instead of assuming that these actions are prescribed in advance as functions of time or as functions of the state of the system; b) multiple interactions between human driven vehicles and autonomous vehicles can be modeled simultaneously and in a computationally tractable way; c) all the vehicles in a traffic scenario are simultaneously modeled as decision makers as opposed to predicting the decisions of one vehicle while assuming that the rest of the vehicles move based on certain kinematic and dynamic constraints. The proposed game theoretic model makes it possible to conduct a quantitative analysis of the traffic. For example, a) the increase in the number of incidents based on the traffic density can be assessed; b) the effect of a certain parameter value in an autonomous vehicle control algorithm on the safety of the vehicle, e.g., quantified by the number of incidents, can be determined; c) various autonomous vehicle control algorithms can be compared quantitatively in a multi-vehicle time-extended scenario, based on certain safety and performance metrics; and d) optimization of a cost function that reflects safety and performance can be performed.

Our approach uniquely combines a specific game theoretic formalism, which is used to model intelligent agent interactions, and reinforcement learning, which is used to evolve these interactions in a time-extended (multi-move) scenario. The core ideas are synergistic with the framework of "semi network-form games," [23], [24] and help us obtain the probable outcomes of a complex traffic scenario driven by multiple interactions. To the authors' knowledge, such an approach has not been previously exploited for automotive traffic modeling.

Other game theoretic approaches, in particular, based on Stackelberg games, have been studied for application to vehicle highway driving problems in [15] and [16]. Although these approaches represent driver interactions in traffic using a game theoretic setting, they do not consider dynamic (multi-move) scenarios. The latter are considered in [25] for Hybrid Electric Vehicle (HEV) energy management where the driver and the powertrain are considered to be two players in a game. However, increasing the number of players (drivers, in our case) beyond three complicates computing a Stackelberg solution, especially in a time-extended (multi-move) scenario. On the other hand, the hierarchical reasoning based game theoretic approach exploited in this paper is easily scalable. Indeed, an implementation of the proposed approach for a 50 player game can be found in [26], and scenarios with up to 30 vehicles are handled in this paper.

To summarize, the contributions of this paper are as follows: a) we develop a novel and scalable to multiple vehicles traffic model and simulation environment based on a specific game theoretic modeling approach; b) our approach allows the representation of driver interactions for many-move, many-vehicle traffic scenarios using a computationally tractable game theoretic approach, whereas many existing methods consider only one-move interaction between two vehicles; c) we demonstrate that autonomous vehicle control algorithms can be quantitatively evaluated in our simulator and compared based on safety and performance metrics applied to simulation outcomes; two policies proposed in the literature serve as case studies, while other autonomous driving algorithms could also be easily integrated and tested; d) we provide a case study to demonstrate that probabilistic outcomes of traffic scenarios can be utilized to obtain the optimal calibration for an autonomous vehicle controller.

Preliminary results have appeared in conference papers [27], [28]. Differently from [27], [28], in this paper, a) we incorporate a more realistic action space including harder brakes and faster accelerations; b) we develop a more realistic traffic model with more representative distance constraint violation rates via improvements in the reinforcement learning procedure; and c) we demonstrate that optimal parameter values for an autonomous vehicle control algorithm can be obtained using a cost function based on safety and performance. Furthermore, this paper has additional details and interpretations not found in our preliminary conference papers.

The paper is organized as follows. In Section II, we define the problem being treated in this paper and the vehicle model being used. In Section III, we present the level-k game theory and the reinforcement learning approach to obtain the policies. In Section IV, we introduce two autonomous vehicle control algorithms that will be tested by our simulator. In Section V, we describe our simulator and its implementation results. We then summarize the key developments in our paper in Section VI.

II. Problem Definition

The problem we treat is to model the behavior of drivers in a traffic scenario where the cars are driven on a 3-lane highway. We later demonstrate that such models can be used in simulators to evaluate autonomous vehicle control policies. Fig. 1 shows an example scenario with 6 cars. Note that this is not a restriction of the proposed method, and that scenarios with more cars and lanes can be handled. Simulated cars are assumed to be traveling in the same direction and to be driven by human drivers who obey the general traffic laws.

A. Physical models

The discrete-time equations of motion for each vehicle during forward motion and lane changes are given by

$$\begin{aligned} x(t+1) &= x(t) + v_x(t)\Delta t, \\ v_x(t+1) &= v_x(t) + a(t)\Delta t, \\ y(t+1) &= y(t) + v_y(t)\Delta t, \end{aligned} \tag{1}$$

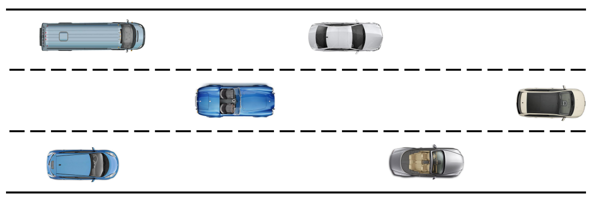
Fig. 1: Example scenario: Traffic in a 3-lane highway.

where x, y are the vehicle longitudinal and lateral positions, respectively, v_x and v_y are vehicle longitudinal and lateral velocities, respectively, $a(t)$ is the vehicle longitudinal acceleration, and Δt is the update period. The longitudinal acceleration $a(t)$ and the lateral velocity $v_y(t)$ are the two control inputs. We assume that all cars accelerate and decelerate at either $\pm a_1$[m/s^2] or $\pm a_2$[m/s^2]. The nominal values, $\pm a_1$[m/s^2], reflect the acceleration/deceleration a human driver would apply in normal situations, while the hard accelerations/decelerations of $\pm a_2$[m/s^2], reflect the values used in urgent situations, based on the maximum acceleration/deceleration capability of a vehicle. We also assume that lane changes occur with constant lateral velocity such that the total time to change lanes is t_{cl}[s]. During lane changes, the longitudinal velocity remains constant, and once a lane change begins, it always continues to completion.

While these assumptions represent a reasonable approximation, formulations with more acceleration/deceleration levels can be similarly introduced at the cost of the increased time and effort to compute the policies.

B. Observation space

In real traffic flow, a driver can neither observe nor process all the information about all cars on the road. A human can possibly observe and use the information he/she obtains from the cars in a certain vicinity of their own. In particular, a human driver can hardly measure his/her exact distances from other cars. He/she can only estimate the distances and specify them as "close","far", etc. Therefore, we assign the following observation space for the drivers:

1) The distance from our car to the car directly in front, called range, quantified as "close" (distance $\leq d_c$[m]), "nominal" (d_c[m] $<$ distance $\leq d_f$[m]) or "far" (distance $> d_f$[m]),
2) The range to the car in the front left lane, quantified as "close", "nominal" or "far,"
3) The range to the car in the front right lane, quantified as "close", "nominal" or "far,"
4) The range to the car in the rear left lane, quantified as "close", "nominal" or "far,"
5) The range to the car in the rear right lane, quantified as "close", "nominal" or "far,"
6) The range rate to the car in front, quantified as "approaching" (distance decreasing), "stable" (distance not changing), or "moving away" (distance increasing),
7) The range rate to the car in the front left lane, quantified as "approaching", "stable" or "moving away,"
8) The range rate to the car in the front right lane, quantified as "approaching", "stable" or "moving away,"
9) The range rate to the car in the rear left lane, quantified as "approaching", "stable" or "moving away,"
10) The range rate to the car in the rear right lane, quantified as "approaching", "stable" or "moving away,"
11) The lane index of the car, quantified as "lane 1", "lane 2" or "lane 3."

Note that this description of the observation space reflects the uncertainty (or noise) present in real life driver observation.

C. Action space

Drivers have 7 basic actions:

1) "Maintain" current lane and speed,
2) "Accelerate" at rate = $a_1[\mathrm{m/s^2}]$, provided velocity does not exceed $v_{\max}[\mathrm{km/h}]$,
3) "Decelerate" at rate = $-a_1[\mathrm{m/s^2}]$, provided velocity is above $v_{\min}[\mathrm{km/h}]$,
4) "Hard Accelerate" at rate = $a_2[\mathrm{m/s^2}]$, provided velocity does not exceed $v_{\max}[\mathrm{km/h}]$,
5) "Hard Decelerate" at rate = $-a_2[\mathrm{m/s^2}]$, provided velocity is above $v_{\min}[\mathrm{km/h}]$,
6) Change lane to the left, provided there is a lane on the left,
7) Change lane to the right, provided there is a lane on the right.

D. Reward function

A "reward function" is a mathematical representation of the goals of a driver. Basic goals of the drivers in real traffic are 1) to not have an accident, such as a car crash (safety), 2) to minimize the time needed to reach the destination (performance), 3) to keep a reasonable headway from preceding cars (comfort) and 4) to minimize driving effort (comfort).

These goals can be reflected in the reward function given as

$$R = w_1 c + w_2 v + w_3 h + w_4 e, \tag{2}$$

where w_i, $i = 1,2,3,4$, are the weights for each term and c, v, h and e represent "constraint violation," "velocity," "headway" and "effort" metrics. In particular, we define a safe zone for each car (a rectangular area that over-bounds the geometric contour of the car with a safety margin) whose boundaries are treated as distance constraints to represent the safety of the car.

The weighting terms, w_i, may change depending on the aggressiveness of the driver, but, intuitively, to avoid distance constraint violation should be of the most importance, i.e.,

$$w_1 \gg w_2, w_3, w_4. \tag{3}$$

The terms, c, v, h, e, are explained below.

c **(constraint violation):** The term c is assigned a value of -1 when a distance constraint violation occurs and a value of 0 otherwise.

v **(velocity):** The term v is assigned the value

$$v = \frac{v_x - v_{\text{nominal}}}{a_1}, \quad v_{\text{nominal}} = \frac{v_{\min} + v_{\max}}{2}. \tag{4}$$

Here the reason for dividing by a_1 is to make this term of the same order of magnitude as the other terms, to facilitate the choice of weights.

h **(headway):** The term h takes the following values depending on the headway distance (the range to the car directly in front):

$$h = \begin{cases} -1 & \text{if} \quad \text{headway} \in \text{“close,”} \\ 0 & \text{if} \quad \text{headway} \in \text{“nominal,”} \\ 1 & \text{if} \quad \text{headway} \in \text{“far.”} \end{cases} \tag{5}$$

e **(effort):** The term e takes the value 0 if the driver's action is "maintain," $e_2 = -5$ if the driver's action is "hard accelerate" or "hard decelerate," and $e_1 = -1$ otherwise. This term discourages the driver from making unnecessary maneuvers. In particular, a higher penalty discourages the driver from unnecessarily applying "hard accelerate" or "hard decelerate." But in the case where another maneuver cannot avoid a constraint violation, the driver would apply "hard accelerate" or "hard decelerate" to keep safe. Note that the ratio between e_1 and e_2 depends on the driver's behavior and could be tuned to match the driving behavior of different human drivers.

E. Constraints

The reward function (2) already reflects the penalty for distance constraint violations, which may be viewed as imposing soft constraints on the control. For some combinations of states and actions that obviously lead to constraint violations, we can also impose hard constraints to avoid the occurrence of such combinations. In particular, we introduce the following hard constraints, which make certain actions unavailable in certain situations:

1) If a car in the left lane is in a parallel position, the controlled car cannot change lane to the left,
2) If a car in the right lane is in a parallel position, the controlled car cannot change lane to the right,
3) If a car in the left lane is "close" and "approaching," the controlled car cannot change lane to the left,
4) If a car in the right lane is "close" and "approaching," the controlled car cannot change lane to the right.

Note that two cars are assumed to be in a parallel position if the safe zones of these two cars intersect in the longitudinal direction. The use of these hard constrains eliminates the clearly undesirable behaviors better than through penalizing them in the reward function, and also increases the learning speed during training.

III. Driver Interaction Model

The driver interaction model developed in this study enables the modeling of driver-driver and driver-autonomous vehicle interactions through the use of hierarchical decision making and reinforcement learning. The model is a "policy," which is a stochastic map from the observation space of the driver to his/her action space (see Section II). In other words, this map assigns a probability distribution over possible actions for every observation. Below, we explain how this model is generated.

A. Hierarchical decision making

The developed interaction model is premised on the idea that intelligent agents (such as drivers) have different levels of reasoning. According to this observation, a level-0 agent does not consider probable actions of other agents that he/she is interacting with but rather behaves reflexively. For example, if a driver makes a hard brake when he/she observes an obstacle on the road, without considering how the car following behind would react to this sudden deceleration, this behavior is referred to as a level-0 behavior and the driver is referred to as a level-0 driver. On the other hand, if this driver assumes that the car following behind is being driven by a level-0 driver, who would make a hard brake in case of an obstacle, which may not be enough to avoid a collision, and therefore decides to make a lane change to avoid that collision, then this driver is referred to as a level-1 driver. Similarly, if a driver assumes that the other drivers are level-1 and takes an action accordingly, this driver is a level-2 driver. Higher level driver behavior can be modeled using the same logic. A detailed explanation of this hierarchical modeling method is given in [29] and [30].

To obtain higher level policies, one needs to start by defining a level-0 policy. There are various ways to do this, such as selecting each possible action with equal probability regardless of the observation, or constructing a very simple policy, which provides a minimally reasonable behavior for a range of observations.

In this study, a level-0 policy is formulated as follows:

$$\text{action}_{l0} = \begin{cases} \text{“Decelerate,”} & \text{if the car in front is} \\ & \text{“nominal” and “approaching,”} \\ & \text{or is “close” and “stable,”} \\ \text{“Hard Decelerate,”} & \text{if the car in front is “close”} \\ & \text{and “approaching,”} \\ \text{“Maintain,”} & \text{otherwise.} \end{cases} \tag{6}$$

B. Reinforcement learning to solve the Partially Observable Markov Decision Process

The problem treated in this paper is a multi-agent decision making problem. We use a reinforcement learning (RL) algorithm to determine the policies for each agent based on the reward function defined in Section II-D. To achieve the maximum reward, the RL algorithm exploits two steps including 1) "policy evaluation," where the state-action pairs are assigned values based on the accumulated rewards they gain, and 2) "policy improvement," where the probability of choosing the actions that have higher reward values are increased. For more details on RL, see [31].

Conventional RL algorithms require the process to be Markov for convergence guarantees. Note that although the underlying dynamics of the highway problem studied in this work is Markov, each agent (driver) can observe only a subspace of the whole state space (see Section II-B) and therefore has to solve a Partially Observable Markov Decision Process (POMDP) problem. In RL literature, the observations are commonly referred to as "messages." In this work we employ the Jaakkola RL algorithm [32], which distinguishes itself from conventional approaches by guaranteeing to converge at least to a local maximum in terms of average rewards, when the problem is of POMDP type. Below, the Jaakkola RL algorithm is summarized. See [32] for further details.

The following steps are followed to obtain the driver policies using Jaakkola RL:

Step 1. Evaluate the value function for the messages $V(m|\pi^t)$ and Q-values for the message-action pairs $Q(m,a|\pi^t)$ associated with the driver policy π at step t, using the following equations:

$$\begin{aligned}
\beta^t_m(m) =&(1-\frac{\chi^t_m(m)}{K^t_m(m)})\gamma^{(t)}\beta^{t-1}_m(m)+\frac{\chi^t_m(m)}{K^t_m(m)},\\
V(m|\pi^t) =&(1-\frac{\chi^t_m(m)}{K^t_m(m)})V(m|\pi^{t-1})\\
&+\beta^t_m(m)[R^t-\bar{R}(\pi^t)],\\
\beta^t_a(m,a) =&(1-\frac{\chi^t_a(m,a)}{K^t_a(m,a)})\gamma^{(t)}\beta^{t-1}_a(m,a)+\frac{\chi^t_a(m,a)}{K^t_a(m,a)},\\
Q(m,a|\pi^t) =&(1-\frac{\chi^t_a(m,a)}{K^t_a(m,a)})Q(m,a|\pi^{t-1})\\
&+\beta^t_a(m,a)[R^t-\bar{R}(\pi^t)],
\end{aligned} \tag{7}$$

where m and a designate "messages" (observations) and actions, respectively, the superscript t refers to the time step, while the subscript (m or a) indicates whether the function is associated with messages or with message-action pairs. The function χ is an indicator function, taking the value 1 if the message/message-action pair is visited and 0 otherwise. The function K represents the number of times a particular message/message-action pair is visited. The positive sequence $\gamma^{(t)}$ represents a discount factor and is converging to 1 in the limit as $t\to\infty$. The R^t is the reward obtained at the current time instant t, while $\bar{R}(\pi^t)$ is the average reward associated with the policy π^t, i.e., if this policy were executed for an infinite time [33].

In the implementation of the above algorithm, the true average reward $\bar{R}(\pi^t)$ is replaced with an estimate, which is computed as an average reward over a past window, making use of the fact that the policy is slowly varying in time.

Step 2. Update the driver policy π, which is a probabilistic mapping from observations (messages) to actions, using the following equation:

$$\pi^{t+1}(a|m)=(1-\varepsilon)\pi^t(a|m)+\varepsilon\hat{\pi}^t(a|m), \tag{8}$$

where $0<\varepsilon<1$ is the learning rate and $\hat{\pi}^t$ is chosen such that $J(m|\hat{\pi}^t)=\max_a\big(Q(m,a|\pi^t)-V(m|\pi^t)\big)$. For any policy $\pi(a|m)$, $J(m|\pi)$ is defined as:

$$J(m|\pi)=\sum\nolimits_a\pi(a|m)\big(Q(m,a|\pi)-V(m|\pi)\big). \tag{9}$$

Note that based on the process defined above, $\hat{\pi}^t$ should be defined in such a way that it has probability 1 for picking the action a that has the highest $Q(m,a|\pi^t)$ value and probability 0 for picking the other actions.

Going through step 1 and 2 at each time t, the driver model, or the converged policy, denoted by π^*, is obtained once the policy converges during the iterative process. The convergence criterion is based on the convergence of the average reward, i.e., change in the average reward within a given number of steps less than a tolerance in absolute value.

C. The role of hierarchical decision making in obtaining driver policies

The process of obtaining driver policies is called "training," where the trained driver is a *learner* and the other vehicles and automation constitute the *environment*. During the training process, the model of the environment is needed to obtain state transitions as a result of driver actions, where the hierarchical decision making approach plays a crucial rule: for the training of a level-k driver policy, all of the traffic but the trained driver are assigned level-(k-1) policies. The process starts with the determination of a level-0 policy (see Section III-A), which represents the lowest level where the drivers do not consider interaction with other drivers and do not explicitly take into account their possible actions. Once a level-0 policy is determined, the RL algorithm is run by assigning level-0 policies to all of the vehicles except the one that is being trained. At the end of the training process, a level-1 policy is obtained. Similarly, while training a level-2 policy, all of the vehicles but the trained vehicle are assigned a level-1 policy. This hierarchical assignment continues until the desired highest level is obtained. In experimental studies [30], it is shown that in human interactions, level-3 players are very rarely encountered and therefore in our results we trained policies up to and including level-2.

IV. Autonomous Driving Control Approaches

The proposed traffic model has been employed to build a simulator to test and evaluate the performance of autonomous driving control algorithms. As specific examples, two autonomous driving approaches, based on Stackelberg policies and decision trees, will be evaluated and compared, using a simulator in which the traffic, other than the controlled vehicle, consists of drivers modeled using our game theoretic policies. In this section, the control algorithms that will be tested are briefly explained and in the next section simulation evaluations are provided.

The Stackelberg policies and the decision tree policies that are compared in this study were originally developed in [10], [15] and [16]. Since these policies were developed under assumptions representing a simpler traffic environment, some necessary modifications were made to let the autonomous vehicle, which will employ these policies, be able to operate in the traffic environment investigated in this study. For example, the originally proposed Stackelberg and decision tree policies consider only lane change actions. To make them more compatible with the test environment, acceleration/deceleration actions are added to their action space. Fig. 2 shows the necessity of this modification, where each rectangle represents a car and the arrows represent both the driving direction and velocity (the longer the arrow, the larger the velocity). In the figure, three yellow cars are in front of the red test car and since the speed of the test car is larger than that of the yellow cars, there is a danger of collision that cannot be resolved only with a lane change.

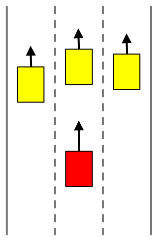

Fig. 2: No solution for the red car by only changing lanes.

A. Stackelberg policies

To generate Stackelberg policies for the autonomous vehicle, we consider three vehicles as players, and the rest of the vehicles are considered to be the environment. The three players are assigned roles as the “leader,” “first follower,” and “second follower,” and they choose their actions sequentially: the leader chooses its action first, followed by the first follower, and, finally, the second follower. Each player evaluates their actions according to a utility function that consists of two parts. The first part, known as the positive utility, is defined as follows:

$$U_{\text{pos}} = \begin{cases} \min(d_{\triangle}, d_v), & \text{if there is a vehicle ahead,} \\ d_v, & \text{otherwise,} \end{cases} \tag{10}$$

where $d_{\triangle}$ is the distance to the car directly in front, i.e., the headway distance, and d_v is the maximum visibility distance. The second part of the utility is known as the negative utility:

$$U_{\text{neg}} = d_{\nabla} - v_r T - d_{\text{min}}, \tag{11}$$

where d_{∇} and v_r are the distance to and the relative velocity of the car immediately behind, T is a prediction time window, and d_{min} is the minimum distance required to allow a lane change; here, d_{min} is set to the car safe zone length. Thus, overtaking vehicles are taken into consideration, and lane changes that cut off overtaking vehicles are discouraged.

The actions chosen by the leader, first follower, and second follower, denoted by γ_ℓ, γ_{f1}, and γ_{f2}, respectively, are the Stackelberg Equilibrium actions; i.e., the leader chooses its actions to maximize its utility for the worst-case actions that the two followers might choose. Thus, the leader chooses:

$$\gamma_\ell^* = \max_{\gamma_\ell} \min_{\gamma_{f1}} \min_{\gamma_{f2}} \left[U'_{\text{pos}} + U'_{\text{neg}} \right], \tag{12}$$

where U'_{pos} and U'_{neg} are the utilities that correspond to a specific set of actions $\{\gamma_\ell, \gamma_{f1}, \gamma_{f2}\}$. The two followers maximize their own utilities with the known choice of γ_ℓ. In this paper, when constructing Stackelberg policies, the controlled vehicle is the leader, and the two cars immediately behind are the followers. An alternative choice in which the controlled vehicle is one of the followers instead of the leader can be treated similarly.

It is noted that in [15] and [16] vehicle dynamics are different than the ones used in this paper. Furthermore, some aspects of the modeling such as uncertainties in measuring distance, side-viewing and response delays that are considered in these references are omitted here to simplify the analysis but can be easily integrated.

B. Decision tree policies

In the decision tree approach to autonomous driving, the vehicle’s actions are determined by a path planner that evaluates a specified number of pre-selected action profiles by building a tree of potential action sequences and evaluating each branch according to a specified metric.

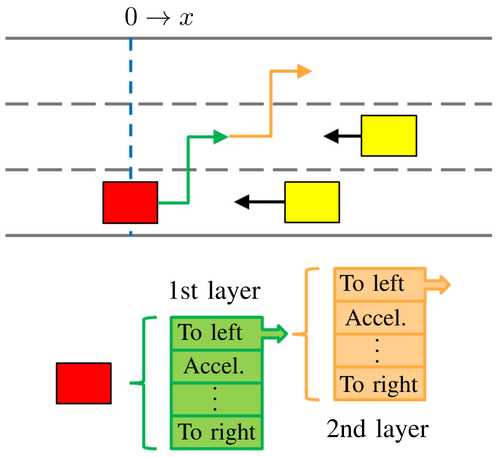

Fig. 3: Decision tree diagram. The black arrows show the relative velocities of the yellow cars with respect to the red test car.

In this paper, the decision tree consists of two layers, where each layer allows the seven actions listed in Section II-C. Thus, the action profiles consist of two actions each, and 49 profiles are evaluated. The evaluation metric is based on the reward function (2), which is also used for the training of the level-k policies. In particular, the total reward is calculated as a weighted sum of the rewards obtained from the two layers:

$$R_{\text{total}} = w_{l1} R_{l1} + w_{l2} R_{l2}, \tag{13}$$

where $w_{l1}, w_{l2} \in \mathbb{R}^+$ are the weighting terms. The car applies the first-layer action of the profile that has the highest total reward among all profiles, and repeats this procedure at each step. When evaluating the action profiles, the controlled vehicle assumes that all other vehicles would apply the action “maintain” during the prediction horizon.

In [10], it was assumed that the environment evolves deterministically over a planning horizon, independently of the controlled vehicle’s actions. However, in our simulator, not only the controlled vehicle responds to the traffic, but also the traffic responds to the controlled vehicle’s actions.

C. Path planner triggering threshold

Both the Stackelberg and the decision tree policies are used as path planners and triggered only when necessary and beneficial. When the policies are not triggered, the vehicle follows a predefined driving pattern. More specifically, with the help of Fig. 4, the driving algorithm can be explained as follows:

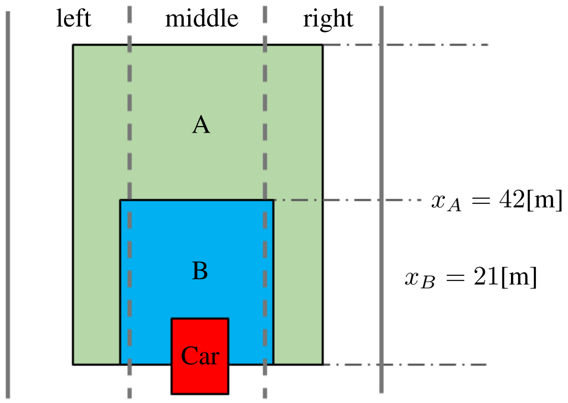

Fig. 4: Triggering threshold.

1) "Accelerate," if there are no other cars in region A,
2) Path planner is triggered, if there are cars in region A, but no cars in region B,
3) "Safe Mode," which in this paper is the same as the level-0 policy, is applied if there are cars in region B.

It is clear that if there are no cars in region A, the autonomous vehicle may accelerate safely, while if there are cars in region B, the autonomous vehicle is supposed to decelerate to keep a reasonable headway distance. The activation logic is designed 1) to increase the safety, and 2) to reduce the computational cost by preventing unnecessary action evaluations. In particular, region A is designed to cover the center of adjacent lanes, while region B is designed to cover the boundary lines of the current lane, so that when some vehicle in the vicinity is changing lanes into the test vehicle's lane, the test vehicle becomes aware of it before the other vehicle actually enters its lane.

V. Results

A. Environment and set-up

We model the environment as follows. The width of a lane is 3.6[m], and the safe zones around the cars, which should not be violated, are modeled as 6[m] $\times$ 2[m] boxes. Cars always drive at the center of a lane unless they are changing lanes. Cars only accelerate or decelerate at $\pm a_1 = \pm 2.5$[m/s^2] or $\pm a_2 = \pm 5$[m/s^2], and lane changes occur with constant lateral velocity such that the total time to change lanes is $t_{cl} = 2$[s]. During lane changes, the longitudinal velocity remains constant and once a lane change begins, it always continues to completion. The longitudinal axis is called x, and its origin is collocated with the car that is to be trained or evaluated.

To configure the simulation, the following values need to be specified:

1) the number of lanes, n_ℓ,
2) the number of cars, n_c,
3) the maximum allowable initialization distance, $x^0_{\max}$,
4) the simulation duration, t_f.

The following procedure is employed to initialize a simulation: A car is assigned to a lane that is determined randomly based on the uniform distribution. The specific location of the car within the assigned lane is determined randomly based on the uniform distribution in $[-x^0_{\max}, x^0_{\max}]$. Then, the initial longitudinal velocity of the car is given randomly based on the uniform distribution within the range $[v_{\min}, v_{\max}] = [62, 98][\text{km/h}]$. The initial action is set as "maintain." The car is then assigned a policy to follow (level-0, 1, or 2). This process is repeated until all cars are configured. While locating each car, it is required that the minimum initial distance between the cars be 30[m]. Once the initialization is completed, the simulation is run according to Algorithm 1 given below.

```
1  $t = 0$.
2  while $t < t_f$ do
3      foreach car do
4          Obtain observations from the environment.
5          Given the observations, determine an action
           based on assigned policy.
6          Given the action, update position and velocity.
7      end
8      if training a policy then
9          Evaluate reward function for trainee.
10         Update value function.
11     end
12     if trainee/test vehicle is in a constraint violation state
       then
13         End the simulation.
14     end
15     $t = t + \Delta t$.
16 end
```

Algorithm 1: Single Episode Simulation

Five cars are observable by the test vehicle, as described in Section II-B, and a car is considered "close" if its relative longitudinal position satisfies $|x_r| \le d_c = 21$[m], "nominal" if $d_c < |x_r| \le d_f = 42$[m], and "far" if $d_f < |x_r| \le d_v = 63$[m], where d_v is the maximum visibility distance. Cars farther away than d_v[m] are considered to be out of visual range and unobservable. If no car can be observed in a position, this is considered equivalent to a car that is "far" and "moving away." Note that $d_c = 21$[m] is determined by considering the minimum distance needed for a car to safely avoid a distance constraint violation by braking with the maximum allowable deceleration in the worst case scenario where a car in front is approaching with maximum relative speed and entering the "close" range.

Fig. 5 shows a snapshot of an example simulation setup with three lanes. The rectangles represent the safe zones around the

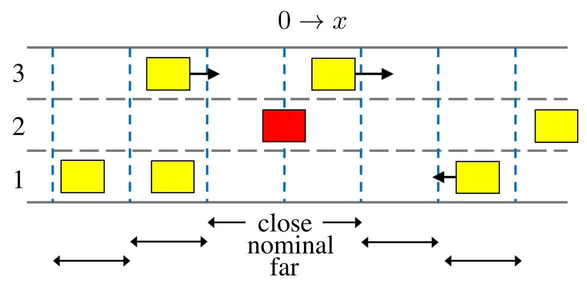

Fig. 5: Simulation environment.

cars, and the arrows attached to the rectangles show the relative velocities of the cars with respect to the test vehicle, that is located in the center lane at $x = 0$. Specific observations by the test vehicle are given as follows:

- Front left: close, moving away,
- Front center: far, moving away,
- Front right: far, approaching,
- Rear left: nominal, approaching,
- Rear right: nominal, stable,
- Lane index: lane 2.

Note that two cars in Fig. 5 are unobservable: The car in the front center position is beyond the visual range – so the corresponding observed status is "moving away" even though the car is actually "stable;" and the car in the rear right "far" position is hidden by the car in the rear right "nominal" position. These limitations in the observation space reflect the POMDP nature of the problem, as discussed previously.

B. Level-0 driver behavior

In this section, we present simulation results to show the driving behavior of a level-0 car. Furthermore, we present the simulator user interface.

In Fig. 6, the red car in the center is the trainee/test vehicle, while the yellow cars make up the traffic environment. The red arrow in front of the red car indicates its travel direction, and arrow size indicates how fast the car is traveling. The panel on the left is a speedometer, and the steering wheel on the right indicates the lateral motion of the car. The green box and red box in the middle indicate the gas pedal and the brake pedal, respectively, and when any of them turns blue, that indicates the pedal is pressed. The coordinate axis is fixed on the test car and the motions of the other cars can be tracked by their relative distance to the red car. In Fig. 6(a), a yellow car directly in front of the red car ("far") is "approaching" because the red car is faster. At this moment neither the gas pedal nor the brake pedal is pressed. In Fig. 6(b), the yellow car enters the "nominal" range, and consistently with the level-0 policy, the red car brakes and decreases its speed. In Fig. 6(c), the red car gets to a lower speed, and the yellow car is now "stable."

C. Training process

When training a new policy, the observation value function, V, for observed message m, and the action value function, Q,

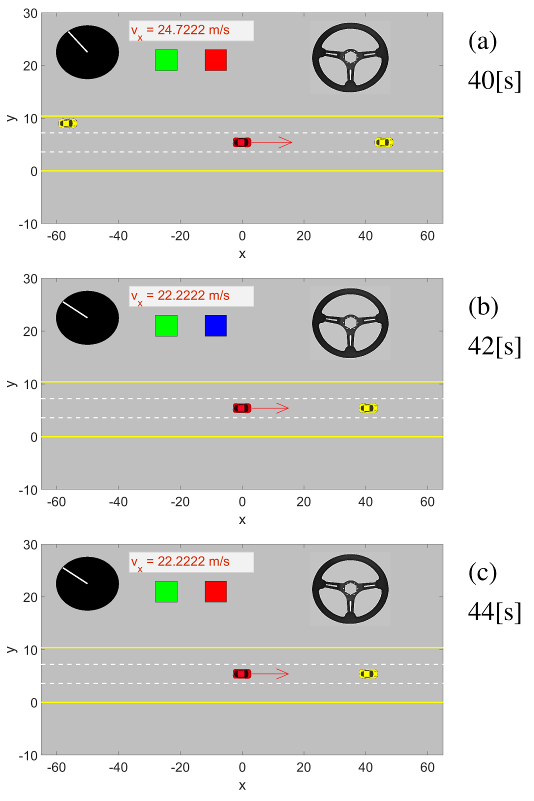

Fig. 6: Level-0 simulation results: Plots (a)-(c) show snapshots of the simulation at 40[s], 42[s] and 44[s], respectively.

for message-action pair (m, a), are initialized as follows:

$$\begin{aligned} &\forall m, \quad V(m) = 0; \\ &\forall m, \forall a, \quad Q(m, a) = 0. \end{aligned} \tag{14}$$

For each observation, the actions are assigned equal probability of selection at initialization, and during each policy improvement step, if

$$\max_a Q(m, a) > V(m), \tag{15}$$

then 0.01 is added to the probability of selecting $\operatorname{argmax}_a Q(m, a)$, after which the action probabilities are normalized.

The observation space described in Section II-B has 3^{11} different observations. In order to ensure that the learning algorithm is exposed to a large portion of the observation space, the trainee needs to be exposed to both sparse and dense traffic environments. Therefore, during training, the number of cars in the environment is selected randomly, based on the uniform distribution, where $0 \le n_c \le n_c^{\max}$. The maximum number of cars, $n_c^{\max}$, is chosen based on the number of lanes and $x^0_{\max}$, such that if $n_c^{\max}$ cars are placed in the environment, the road is near full capacity.

Finally, after sufficient training time, the level-0 policy is assigned to the observations (messages) that are still not visited enough during training, so that conservative actions are performed in such rarely encountered cases.

Training then proceeds according to Algorithm 2, given below:

```
1 step=0;
2 while $step < desired\ training\ cycles$ do
3     Randomly select $n_c \in [0, n_c^{\max}]$.
4     Initialize all cars with level-($k$-1) policies.
5     Evaluate the level-$k$ policy using Algorithm 1.
6     Improve the policy.
7     step=step+1.
8 end
9 Assign level-0 policy to the not well trained cases.
```

Algorithm 2: Training Process

Fig. 7 shows the evolution of the average reward during level-1 and level-2 trainings. The reward function weights are chosen as: $w_1 = 10000$, $w_2 = 5$, $w_3 = w_4 = 1$.

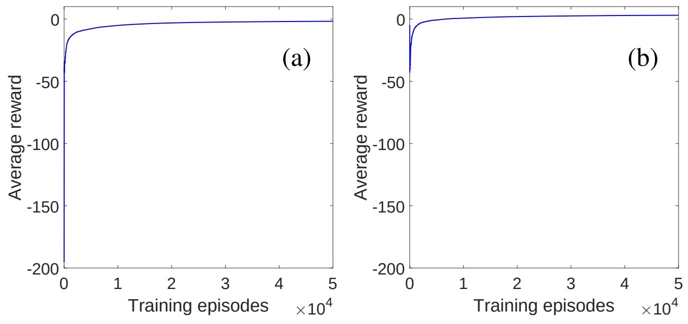

Fig. 7: Average rewards: (a) Training for level-1. (b) Training for level-2.

D. *Level-k driver interactions*

We first present simulation results to show the driving behavior of a level-1 car in a level-0 traffic environment (Fig. 8), and then of a level-2 car in a level-1 traffic environment (Fig. 9). It is noted that in both figures, the traffic is moving towards the right. The test vehicle is red and the rest of the vehicles are yellow.

In Fig. 8(a), the car directly in front of the test vehicle is "approaching" with a large relative speed and enters the "far" distance range. In Fig. 8(b), the red car decelerates and begins to steer. In Fig. 8(c), the red car starts to move into the lane on its left. In Fig. 8(d), the lane changing is completed, after 2[s]; while another car in front approaches the red car. In Fig. 8(e), the red car decelerates until being able to keep a stable headway, which is shown in Fig. 8(f). All these actions represent a reasonable driving behavior.

Similarly, Fig. 9 shows a simulation result of a level-2 car in a level-1 traffic environment. In Fig. 9(a), the yellow car in the middle lane is "approaching" the red car, while the yellow car in the left lane is "moving away," so the red car decides to change lane to the left. In Fig. 9(b), the yellow car in the middle lane starts to change lane to the left, so the red car needs to brake. In Fig. 9(c), there is no car directly in front of the red car, in which case a level-1 car would accelerate. However, because the longitudinal distance of the red car to the yellow car in the middle lane is too small and the red car has no confidence that the yellow car won't change lane to the left, in which case the red car's acceleration would lead to a

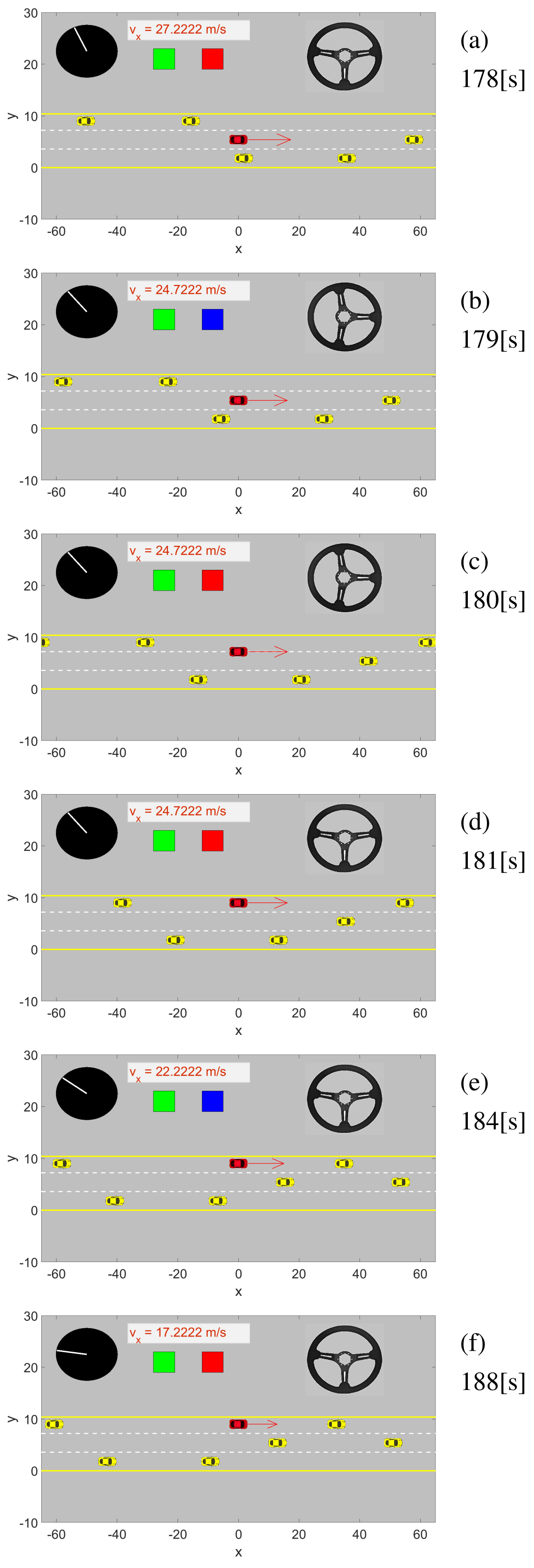

Fig. 8: Level-1 vs. level-0 simulation results: Plots (a)-(f) show snapshots of the simulation at 178[s], 179[s], 180[s], 181[s], 184[s], 188[s], respectively.

distance constraint violation, the red car decides to maintain its current speed. In Fig. 9(d), there is a car in the traffic moving into the red car's lane, which forces the red car to decelerate.

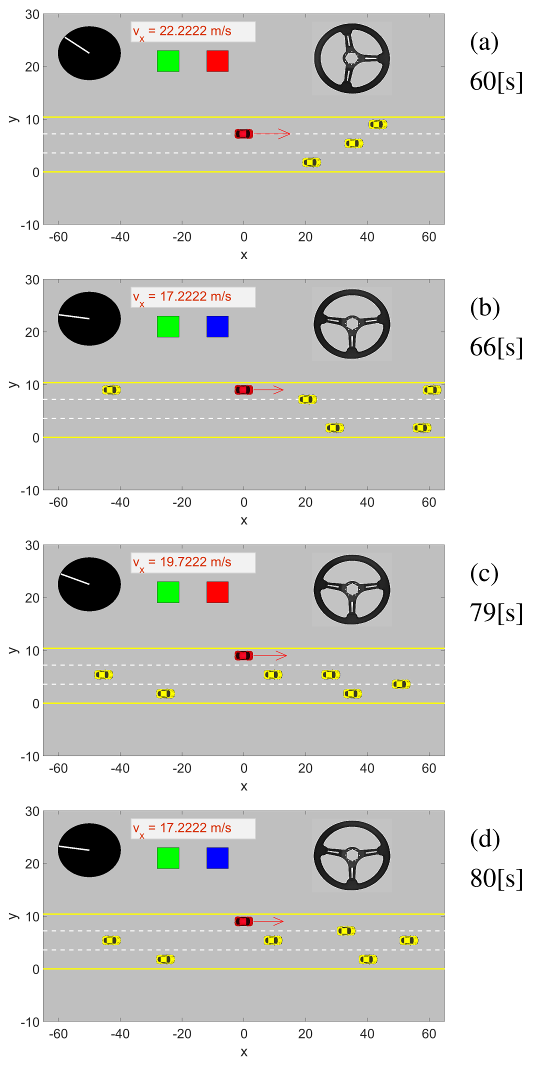

Fig. 9: Level-2 vs. level-1 simulation results: Plots (a)-(d) show snapshots of the simulation at 60[s], 66[s], 79[s] and 80[s], respectively.

Different driving patterns of level-1 and level-2 cars, in similar situations, reflect different levels of reasoning in decision making. It is noted that in Fig. 9, the cars in the environment (in yellow) also change lanes, unlike the ones in Fig. 8, because their actions are based on the level-1 policy, which makes them react to their own surroundings.

Fig. 10 shows the constraint violation rates of the level-1 and level-2 test cars for varying numbers of cars in the traffic. Here, "constraint violation" refers to the violation of the safe zone of the test car by any of the vehicles in the simulations. To obtain these rates, 10,000 simulations are run for each case (i.e., for each value of the number of cars). Each simulation lasts 200 seconds and the rates are provided as the percentage of simulation runs during which safe zones are violated. Note that the level-2 test car is placed in traffic of level-1 vehicles, and the level-1 car is placed in the environment of level-0 vehicles. It is seen that the level-2 test car experiences higher violation rates than the level-1 test car. One explanation for this is that the traffic flow consisting of level-1 cars, where the level-2 policy is evaluated, is much harder to predict compared to the one consisting of level-0 cars, where the level-1 policy is tested.

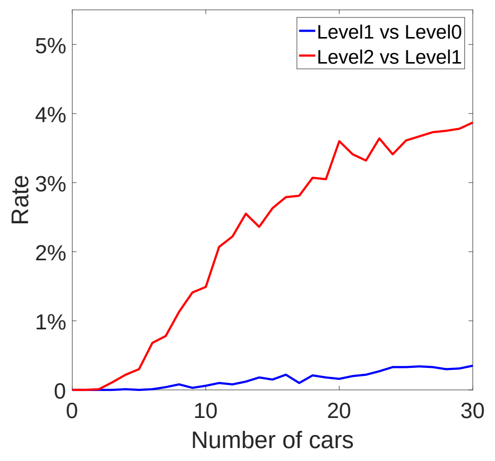

Fig. 10: Constraint violation rates of the level-1 and level-2 policies.

E. A comparative quantitative evaluation of Stackelberg and decision tree policies

At first, we test the Stackelberg and the decision tree policies in a traffic consisting of only level-0 vehicles. A defining feature of the level-0 policy is that the vehicles do not change lanes. It is observed that both the Stackelberg and the decision tree policies perform well in this environment, i.e., constraint violations are not observed. This is also in agreement with the results in [15], [16] and [10]. The figures are omitted as they provide no additional information.

Next, we consider a simulated traffic environment where 10% of the drivers make decisions based on level-0 policies, 60% of the drivers act based on level-1 policies and 30% use level-2 policies. These percentages of various levels are assumed based on an experimental study conducted in [30]. Fig. 11 shows the distance constraint violation rates for the Stackelberg and decision tree policies vs. the number of cars in the traffic. Each simulation is 200 seconds long, and 10,000 simulations are run for each case (i.e., for each value of the number of cars).

As seen in Fig. 11, both approaches exhibit significant distance constraint violation percentages. Note that these violations could also be caused by our level-k drivers, but as shown in Fig. 10, compared to the numbers in Fig. 11, the constraint violation rates of the level-k policies are small.

The main reason for the distance constraint violations is that the developed traffic model with interacting drivers is more complex than the traffic models used for the development of the Stackelberg and decision tree algorithms. To the best of our knowledge, few works addressing the development of

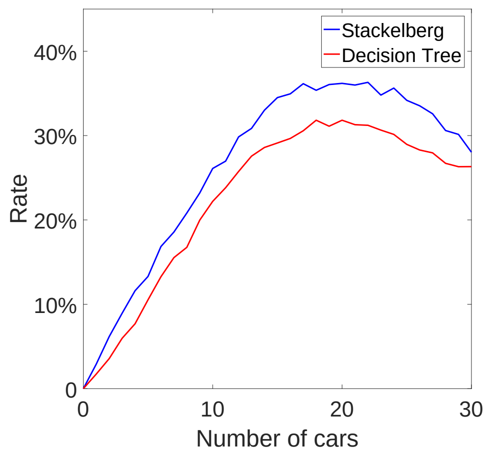

Fig. 11: Constraint violation rates of the Stackelberg and decision tree policies.

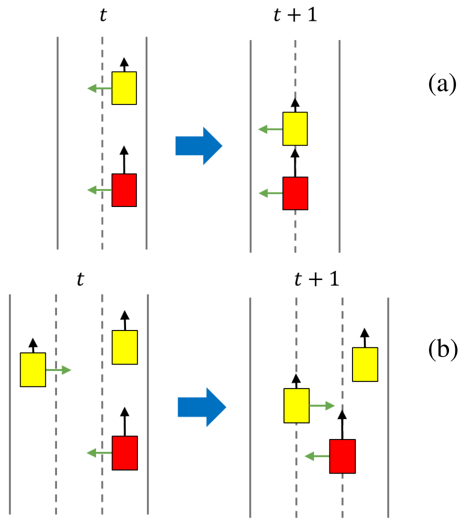

Fig. 12: Scenarios for distance constraint violation.

autonomous vehicle control policies consider traffic of similar complexity.

Fig. 12 shows two cases of driver interactions that are responsible for many distance constraint violations. The red rectangle indicates the car being tested, while the yellow rectangles are the cars in the traffic environment. The black arrows indicate the longitudinal velocities of the cars, while green arrows indicate the lateral velocities. The left hand side presents the scenarios at time t, while the right hand side presents the scenarios at time $t+1$. In Fig. 12(a), the red car is starting to change lane to the left, trying to overtake the front car, while at the same time the front car is also starting to change lane because of some other car ahead of it (not shown in this figure). As a result, both cars are changing lanes while their longitudinal distance keeps decreasing until below the safe distance constraint. Although the red car may brake hard after the lane change, trying to avoid a distance constraint violation, at that time its range is already too small for it to avoid that by braking. In Fig. 12(b), the red car is starting to change lane to the left to overtake the car in front of it. Although the car in front remains in its own lane, there is some other car in the leftmost lane starting to move into the middle lane as well. As a result, both the red car and the yellow car in the leftmost lane are changing lanes to the middle and eventually violate the safe distance between them.

We note that challenging scenarios as the ones above can greatly facilitate the testing of autonomous driving control algorithms. In fact, the above two cases are also dangerous situations for a human driver. Many traffic accidents result from the driver misjudging the potential actions of the surrounding vehicles.

Note also that the constraint violation rate first increases and then decreases as the number of cars in the environment, which reflects the traffic density, increases. As Fig. 13 shows, when the traffic is very sparse, cars on the road can drive almost freely and have low chance of having a constraint violation (①). As the number of cars in the traffic increases, the chance of experiencing an incident also increases (②) (until a peak ③). When the traffic becomes very dense, for instance, in a traffic jam, the average traveling speed becomes low, and at the same time each car mostly stays in its own lane and has few lane change maneuvers. As a result, the probability of having constraint violations becomes low again (④).

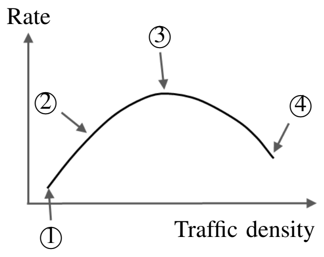

Fig. 13: Diagram of constraint violation rate.

Apart from using the constraint violation rate as a metric to measure the safety and robustness of the Stackelberg and decision tree policies, we also use the average travel speed to measure the driving performance. It can be observed from Fig. 14 that the decision tree policy has better driving performance compared to the Stackelberg policy.

Fig. 15 compares the computational load of the two autonomous driving control methods. The numbers shown in the plot are the average time consumed to run the 200[s] simulation. Because the decision tree policy needs to evaluate 49 profiles of 2-layer actions in total, its required computation load is higher than that of the Stackelberg policy. As a summary, our implementation of the decision tree algorithm is better than our implementation of the Stackelberg algorithm in terms of both safety and performance, while the price paid is the higher computational cost. The numbers in Fig. 15 are obtained using the Java $System.nanoTime()$ function, and the simulations are run on a desktop with i7-4790 3.60GHz CPU and with Eclipse Java Neon platform. Note also that as Fig. 15 indicates the simulator is quite fast.

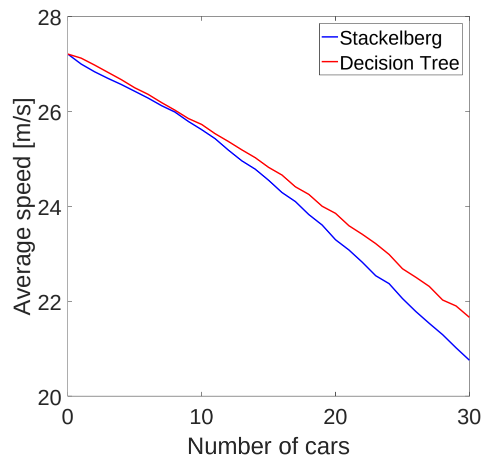

Fig. 14: Average traveling speed of the Stackelberg and decision tree policies.

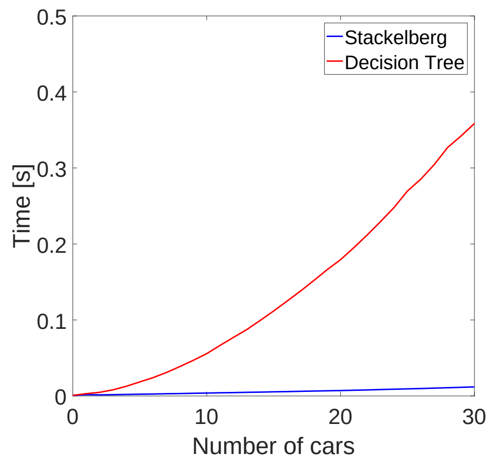

Fig. 15: Computational time of the Stackelberg and decision tree policies.

F. Optimal autonomous driving controller calibration

One of the potential uses of the proposed traffic modeling approach is for calibrating parameter values in the autonomous driving control algorithms. We illustrate this using the decision tree policy as an example.

The optimization objective is defined by considering both safety and performance: the goal is to maximize the following reward function:

$$R_{\mathrm{obj}} = p_1(-\bar{c}) + p_2 \frac{\bar{v}_x - v_{\min}}{v_{\max} - v_{\min}}, \tag{16}$$

where the weights p_1 and p_2 are determined by the user, $\bar{c}$ is the constraint violation rate defined as in Fig. 11, $\bar{v}_x$ is the average speed during the 200[s]-simulations, and $v_{\min}$ and $v_{\max}$ are the lower and the upper bounds of the test vehicle's speed, respectively. Note that this reward function is designed such that each of its terms is dimensionless. The parameters optimized in this example are the ratio $\frac{w_{l1}}{w_{l2}}$, representing the weighting of the two layers in the evaluation metric function (13) in the decision tree evaluations, and x_B, the size of region B in the longitudinal direction – a threshold of triggering the path planner. These two parameters are selected for optimization since they have indirect influence on safety and performance, making them difficult to set from intuition. Note that the influence of other parameters, for example, $w_1, \cdots, w_4$, in the decision tree evaluation metric function, is more intuitive and they can be tuned more easily.

Fig. 16 shows the surface of the reward values as a function of $\frac{w_{l1}}{w_{l2}}$, x_B corresponding to different weight selections p_1, p_2, in the presence of a 20-car traffic. These figures can be used to pick the best pair of $(\frac{w_{l1}}{w_{l2}}, x_B)$ for a given reward function. For example, for maximum safety ($p_1 = 1$, $p_2 = 0$), Fig. 16(a) shows that the best pair is $(2.5, 23)$. The rate of constraint violation with this pair is 27.5%; while for the original selection $(2, 21)$ in the previous section, the corresponding violation rate is 31.8%.

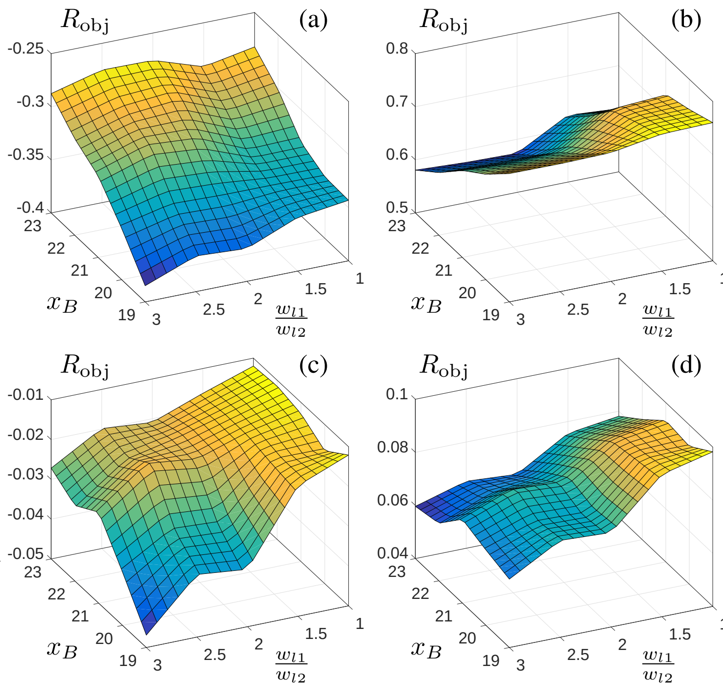

Fig. 16: Parameter optimization results corresponding to different reward function designs. (a) p_1=1, p_2=0, (b) p_1=0, p_2=1, (c) p_1=0.7, p_2=0.3, (d) p_1=0.6, p_2=0.4.

VI. Summary

In this paper, a hierarchical reasoning game theory based approach to model interacting driver behavior in traffic was presented. The proposed method provides an approach to simulate interactive driver behavior under the given traffic conditions.

A traffic simulator was developed using level-k driver models. It can be used for testing and verification of autonomous driving algorithms, and for discovery of challenging trajectories and scenarios that can facilitate the testing of future autonomous vehicles. To illustrate the simulator use, we have defined and tested two autonomous vehicle control policies in terms of safety and performance. Our traffic simulator can also be used for parameter calibration of these policies by a simulation-based optimization approach.

REFERENCES

[1] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[2] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, "Autonomous vehicle technology: A guide for policymakers," RAND Corporation, Santa Monica, CA, Research Report, 2016. Available at http:www.rand.orgpubsresearch_reportsRR443-2.html.

[3] N. Kalra. (2016, Feb. 01), "Australian innovation," [Weblog entry], *THERANDBLOG*, Available: http:www.rand.orgblog201602with-driverless-cars-how-safe-is-safe-enough.html [Apr 03, 2016].

[4] T. Wongpiromsarn and R. M. Murray, "Formal verification of an autonomous vehicle system," in *Conference on Decision and Control, May*, 2008.

[5] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. Lamperski, "Periodically controlled hybrid systems: Verifying a controller for an autonomous vehicle," in *Proceedings of the HSCC: Hybrid Systems, Computation and Control Conference*. California Institute of Technology, 2008.

[6] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, 1998.

[7] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[8] A. Carvalho, S. Lefevre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty-a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[9] I. Miller, M. Campbell, D. Huttenlocher, F.-R. Kline, A. Nathan, S. Lupashin, J. Catlin, B. Schimpf, P. Moran, N. Zych, E. Garcia, M. Kurdziel, and H. Fujishima, "Team cornell's skynet: Robust perception and planning in an urban environment," *Journal of Field Robotics*, vol. 25, no. 8, pp. 493–527, 2008.

[10] L. Claussman, A. Carvalho, and G. Schildbach, "A path planner for autonomous driving on highway human mimicry approach with binary decision diagrams," in *Proceedings of the European Control Conference*, Linz, Austria, July 2015.

[11] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *Proc. IEEE Intelligent Transportation Systems (ITSC)*, October 2014, pp. 392–399.

[12] E. Galceran, A. G. Cunnigham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, 2015.

[13] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[14] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using linearization approach," in *Proceedings of the 16th IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, October 2013.

[15] J. H. Yoo and R. Langari, "Stackelberg game based model of highway driving," in *Proc. ASME Dynamic Systems and Control Conference joint with JSME Motion and Vibration Conference*, Fort Lauderdale, Florida, Oct. 2012.

[16] ——, "A stackelberg game theoretic driver model for merging," in *Proc. ASME Dynamic Systems and Control Conference*, Palo Alto, California, Oct. 2013.

[17] S. Lefevre, Y. Gao, D. Vasquez, E. Tseng, R. Bajcsy, and F. Borrelli, "Lane keeping assistance with learning-based driver model and model predictive control," in *Proceedings of the 12th international symposium on advanced vehicle control*, 2014.

[18] S. Lefevre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *IEEE Intelligent Vehicles Symposium*, 2015, pp. 920–926.

[19] R. Vasudevan, V. Shia, Y. Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli, "Safe semi-autonomous control with enhanced driver modeling," in *Proc. Amer. Control Conf.*, 2012, pp. 2896–2903.

[20] V. Shia, Y. Gao, R. Vasudevan, K. D. Campbell, T. Lin, F. Borrelli, and R. Bajcsy, "Semiautonomous vehicular control using driver modeling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2696–2709, 2014.

[21] D. Salvucci, E. Boer, and A. Liu., "Toward an integrated model of driver behavior in cognitive architecture," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1779, pp. 9–16, 2001.

[22] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5, pp. 351–371, 2002.

[23] R. Lee and D. Wolpert, *Chapter: Game theoretic modeling of pilot behavior during mid-air encounters.* in Decision making with multiple imperfect decision makers. Intelligent Systems Reference Library Series. Springer, 2011.

[24] S. Backhaus, R. Bent, J. Bono, R. Lee, B. Tracey, D. Wolpert, D. Xie, and Y. Yildiz, "Cyber-physical security: A game theory model of humans interacting over control systems," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2320–2327, 2013.

[25] C. Dextreit and I. V. Kolmanovsky, "Game theory controller for hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 652–663, 2014.

[26] Y. Yildiz, A. Agogino, and G. Brat, "Predicting pilot behavior in medium-scale scenarios using game theory and reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1335–1343, 2014.

[27] D. Oyler, Y. Yildiz, A. Girard, N. I. Li, and I. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *Proc. Amer. Control Conf.*, Boston, MA, 2016.

[28] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems," in *IEEE Conference on Decision and Control*, Las Vegas, accepted for publication, 2016.

[29] D. Stahl and P. Wilson, "On players models of other players: Theory and experimental evidence," *Games and Economic Behavior*, vol. 10, no. 1, p. 218254, 1995.

[30] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing models of strategic thinking in Van Huyck, Battalio, and Beil's coordination games," *Journal of the European Economic Association*, vol. 7, no. 2-3, pp. 365–376, 2009.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge: MIT press, 1998.

[32] T. Jaakkola, P. S. Satinder, and I. Jordan., "Reinforcement learning algorithm for partially observable markov decision problems," *Advances in Neural Information Processing Systems 7: Proceedings of the 1994 Conference*, 1994.

[33] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms, and empirical results," *Machine learning*, vol. 22, no. 1-3, pp. 159–195, 1996.