2020 American Control Conference
Denver, CO, USA, July 1-3, 2020

Receding Horizon Motion Planning for Automated Lane Change and Merge Using Monte Carlo Tree Search and Level-K Game Theory

Shahab Karimi* and Ardalan Vahidi

Abstract— **Motion planning and predicting the future states of the surrounding environment are among the main challenges in automated driving. In lane change and merge maneuvers, it is important to know how neighboring vehicles will react in the imminent future. Such a problem becomes more demanding in the absence of inter-vehicular communication (such as V2V, V2X, etc.). Human driver models, probabilistic approaches, rule-based techniques, and machine learning methods have addressed this problem only partially as they do not focus on the behavioral features of the vehicles. In addition, the framework that undertakes the prediction is expected to be fast in providing the path planner with the estimate of future states of the vehicles. Constructing such a fast structure, considering interactions between vehicles, is the main motivation of this study. In this paper we present a fast receding horizon algorithm based on Monte Carlo tree search for real-time path planning in highway scenarios. Inspired by recent results in [1] [2], we adopt a level-k game framework for predicting the strategy of the neighboring vehicles. Our simulations show promising results with fast computations.**

I. INTRODUCTION

Recently, lane change and merge maneuvers have gained much attention in the field of automated driving. The main challenge is to perform a maneuver safely, comfortably, and with least impact on traffic flow. The problem is challenging because future intent of surrounding vehicles is often unknown and the path planning is computationally demanding due to its often nonconvex feasible domain and many hard and soft constraints that need to be considered [3]. The main goal of this paper is to develop a real-time implementable algorithm for motion planning that considers interactions with neighboring vehicles using game theory.

In scenarios where multiple vehicles might collide with each other, a successful trajectory results in no collision while could also consider energy efficiency and duration of the maneuver. In a receding horizon approach, more information of the surrounding environment could be utilized to increase the safety and optimality of the planned trajectory. A common approach to gain such information is possible through some form of communication between the agents [3][4]. Although the mentioned approach has demonstrated effective results in automated driving, loss of the communication, delays or cyber attacks might result in an undesired situation. More importantly, in the foreseeable future it is unlikely that surrounding vehicles be all connected vehicles. In absence of communication, analytical models of human driver behavior, such as MOBIL [5], or probabilistic models [6] can be used. Machine learning and artificial intelligence are also useful techniques for path planning in presence of other agents [7][8][9]. However, all mentioned methods become challenging in terms of the complexity, computational cost, and the large datasets needed for training and validation.

To reduce the computational complexity, in this paper we propose to use Monte Carlo tree search (MCTS) to find a close-to-optimal trajectory for a vehicle in highway driving scenarios. Inspired by recent results in [1], we utilize level-k game theory to predict the trajectory of surrounding vehicles based on their assumed depth of strategizing. In [2] and [10], Li et al. apply level-k approach along with reinforcement learning to create a framework for predction and motion planning in uncertain environments. A similar approach is applied in [1] and [11] for other scenarios such as unsignalized intersection.

Monte Carlo tree search (MCTS) is a random-sampling search technique that emerged by integrating the Monte Carlo method in game-tree search [12]. MCTS has been utilized for strategic games like Ms Pac-Man [13], game of Go [14], and Kriegspiel [15] where the environment (i.e. state space) is partially observable. Such a condition causes uncertainties in desicion making. Moreover, it becomes more challenging in case of need for a fast decision in such a environment. MCTS simulates a large number of possible policies and evaluates their outcome systematically. And eventually, selects the best action to be executed by the agent. In [16], Browne et al. explain the profound influence of this technique in AI methods and planning problems.

In a multi-agent environment where the agents are selecting the optimal sequence of actions (i.e. policy) to fulfill certain desires, conditions might occur where each agent's action would influence the others. For instance, in highway driving scenarios, collision between the vehicles is a major concern. Predicting the sequence of actions that each involved agent could choose over a horizon of time and incorporating such prediction in the sampling process of the ego vehicle, could reduce the chance of conflicting interactions. In the integration of the level-k game theory, in principle, each path planner assumes a lower level of sophistication for each interacting driver and employs MCTS to predict and evaluates their probable sequence of actions over the horizon. Ultimately, the planner executes another MCTS to find its own constraint-admissible trajectory based on the predicted actions of other agents.

In the rest of this paper, Section II. A. describes the

*Corresponding author

Shahab Karimi (skarimi@clemson.edu) and Ardalan Vahidi (avahidi@clemson.edu) are with the Department of Mechanical Engineering, Clemson University, Clemson, SC 29634-0921, USA.

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.

dynamic model that was utilized to capture the dynamics of vehicles along with the actions space. In Section II. B., the reward function is explained. Then, a description of the MCTS is presented in III and Section IV ellaborates on the level-k game theory and its integration within MCTS. Section V presents the simulation results following by conclusions in Section VI.

II. MODELING FRAMEWORK

A. Dynamic model and action space

As this study is mostly focused on developing a high-level controller for automated driving in highways, a simplified kinematic model is used to capture the motion of vehicle(s) in a highway. The discretized state space model is given by

$$\begin{aligned} x_{k+1} &= x_k + v_k.\cos(\theta_k).\Delta t \\ y_{k+1} &= y_k + v_k.\sin(\theta_k).\Delta t \\ v_{k+1} &= v_k + a_k.\Delta t \\ \theta_{k+1} &= \theta_k + \omega_k.\Delta t \end{aligned} \tag{1}$$

where x and y are the longitudinal and lateral position, v and a are the longitudinal velocity and acceleration, θ is the yaw angle and ω is the yaw rate. The control inputs (actions) to this model are given by (a, ω) as determined by the planner. The actions are selected from a discretized action space that is defined as follows:

TABLE I
ACTION SPACE, DETAILS AND DESCRIPTION OF ACTIONS

#	Description/ combination	Acceleration (m/s^2)	Yaw rate (rad/s)
1	maintain	0.00	0.00
2	low brake	-1.50	0.00
3	low accelerate	1.50	0.00
4	mid brake	-3.50	0.00
5	high accelerate	2.50	0.00
6	high brake	-5.00	0.00
7	low left steer	0.00	π / 4
8	low right steer	0.00	-π / 4
9	high left steer	0.00	π / 2
10	high right steer	0.00	-π / 2
11	acc. + left str.	1.50	π / 4
12	acc. + rightstr.	1.50	- π / 4
13	brk. + left str.	-1.50	π / 4
14	brk. + right str.	-1.50	-π / 4

B. Reward function

During the receding horizon planning of a maneuver, at each step, the algorithm determines the state by (1). Then, it computes the action's instantaneous reward by a function of states of the ego vehicle and the traffic that is given by

$$\begin{aligned} r(a_k, s_k) =& \\ &\pi_1.\Gamma_{\text{collision}} + \pi_2.\Gamma_{\text{safe dist.}} + \pi_3.\Gamma_{\text{off-road}} \\ &+\pi_4.\Gamma_{\text{bet. lines}} + \pi_5.\Gamma_{\text{speed}} + \pi_6.\Gamma_{\text{yaw}} + \pi_7.\Gamma_{\text{decel.}} \end{aligned} \tag{2}$$

where Γ denotes the seven elements which we chose to represent the measures for constraint violation and π denotes the weights assigned to each based on their importance. The cumulative reward of the simulated policy, over the planning horizon is determined by

$$\Re(\wp_i) = \sum_{k=0}^{m-1} d^k.r_i(a_k, s_k) \tag{3}$$

where m is the length of the planning horizon, $\wp_i$ refers to the simulated policy by MCTS for the i^{th} agent, and the importance of the instantaneous actions toward the end of the horizon is discounted by $d\in(0,1)$. Each Γ in (2) is explained in detail below. Note that each Γ is scaled in the [0,1] interval which eases tuning of the weights in the algorithm's script.

1) $\Gamma_{\text{collision}}(s_i, S'_i)$: A piecewise function of the state of the agent (s_i) and the state of the traffic (S'_i) which returns 1 if, after deploying the action, the agent does not collide with any other object/agent and returns 0 otherwise. In other words, this function rewards the action which does not result in a crash.

2) $\Gamma_{\text{safe dist.}}(s_i, S'_i)$: Another piecewise function of both s_i and S'_i which returns 1 if the specified safe boundary of the agent does not overlap with any other agent's and returns 0 otherwise.

3) $\Gamma_{\text{off-road}}(s_i)$: A function of only the agent's state (s_i) which returns 1 if the agent's body does not overlap with the off-road region and returns 0 otherwise.

4) $\Gamma_{\text{bet. lines}}(s_i)$: This function returns 1 if the agent remains between highway's dashed lane markers after the executed action and returns 0 otherwise.

5) $\Gamma_{\text{speed}}(s_i)$: This function rewards maintaining the speed as close as possible to the desired speed ($v_{\text{dsrd.}}$) and is given by

$$\Gamma_{\text{speed}}(s_i) = \begin{cases} 1 & \text{if } |v - v_{\text{dsrd.}}| \le 1 \\ 0 & \text{if } v_{\text{dsrd.}} < |v - v_{\text{dsrd.}}| \\ 1 - \frac{|v - v_{\text{dsrd.}}|}{v_{\text{dsrd.}}} & \text{otherwise} \end{cases} \tag{4}$$

6) $\Gamma_{\text{yaw}}(s_i)$: A function that rewards maintaining the vehicle heading along the longitudinal direction, and is given by

$$\Gamma_{\text{yaw}}(s_i) = \begin{cases} 1 & \text{if } |\psi| \le 0.01 \\ 1 - \frac{4}{\pi} & \text{if } 0.01 < |\psi| \le \frac{\pi}{4} \\ 0 & \text{if } \frac{\pi}{4} < |\psi| \end{cases} \tag{5}$$

7) $\Gamma_{\text{decel.}}(s_i)$: A function which returns 0 if the vehicle decelerates in absence of nearby traffic. Nearby traffic is an observation state and is defined as existence of any vehicle within a predefined vicinity ahead of the ego. This function returns 1 (i.e. rewards the action) if either no deceleration occurs in the absence of traffic or the vehicle decelerates while approaching slow traffic.

III. MONTE CARLO TREE SEARCH

Merging and lane changing maneuvers resemble a game where two or more players interact with each other. At each state of the game, which is called as a "node" in the game theory literature, the player tries to exploit an action to maximize the resultant reward. The MCTS algorithm performs numerous virtual policy simulations iteratively ahead of launching the actual policy at each step of the maneuver. The algorithm computes and remembers the cumulative reward of each simulated (or virtual) policy. At each iteration, MCTS utilizes the mentioned reward in a systematic way to select

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.

the virtual policy. This innovative feature is an important distinguishing property of MCTS. Relying on the "Law of large numbers", the MCTS algorithm converges to the policy that is expected to result in the highest cumulative reward. At each state of our specific problem, the action is chosen from the actions presented in Table I. Thus, to find the best policy over the horizon prior to its execution at each actual step of the maneuver, 14^m different combinations of actions must be evaluated in a brute force approach. This is a large number even for relatively short horizon lengths. Even when using Bellman's principle of optimality, the number of reduced computations is still high. For instance, when each of the four states is quantized to n_x values the number of stage reward evaluations is going to be $(n_x)^4 \times m \times 14$. Even with a coarse quantization of $n_x = 10$ and with a prediction horizon of $m = 10$, we are looking at 1.4 million evaluation of reward function for each interacting agent. In such case, the prediction of neighboring vehicles' trajectories following the ego's trajectory planning could translate to a high computational cost. In other words, time complexity and space complexity would be high for an algorithm that is expected to find the best policy even over a not-so-populated action space for a mid-length horizon. Therefore, the application of Monte Carlo tree search, as a systematic and heuristic mechanism, with respect to the law of large numbers is a possible solution to overcome the computational issues, albeit when we can forego optimality.

The recursive process of MCTS consists of four main steps that are described below briefly. More detailed description can be found in [12] and [16].

1) Selection: it refers to the operation of selecting actions from the action space to form a policy to be simulated. This step of MCTS follows two selection patterns referred to as **tree policy** and **default policy**. Tree policy operates only among the existing nodes of the tree. During each iteration, tree policy starts from the root node (i.e. the current state of the MCTS agent), and traverses through the search tree and selects actions striking a balance between 1) exploitation of relatively high rewarding actions, and 2) exploration of the relatively less selected actions. Default policy triggers when the tree policy selects an un-simulated node that is not a terminal node (i.e. the last step of the receding horizon). Default policy selects a sequence of actions randomly for the remaining portion of the horizon. Assume an MCTS algorithm is running for decision making over a horizon of length m among the n_α actions. Following the tree policy, the algorithm selects the action sequence, for instance given by $\wp_{\text{tree}} = \{a_2^0, a_9^1, \ldots, a_5^{m-l-1}, a_3^{m-l}\}$, where the subscript is the index of action in the action space and the superscript is the index of action in the sequence which is less than m. In other words, in this example, a_3^{m-l} is an un-simulated leaf node. Leaf node refers to a node without any child node. From this point of the horizon to the end, the default policy selects actions till the terminal state is reached and outputs the randomly selected sequence, for instance given by $\wp_{\text{default}} = \{a_7^{m-l+1}, a_1^{m-l+2}, \ldots, a_2^{m-2}, a_0^{m-1}\}$. The output of this step is a sequence of actions for the receding horizon given by

$$\wp = \wp_{\text{tree}} \cup \wp_{\text{default}} \tag{6}$$

The procedure of the tree policy selection employs the upper confidence bounds for trees (UCT) for balancing exploration and exploitation [16]. Different UCT functions have been studied within the scope of MCTS. Its most common format is given by

$$UCT = \frac{r_a}{n_a} + c.\sqrt{\frac{\log N}{n_a}} \tag{7}$$

where N is the current total number of MCTS simulations, n_a is the total number of the simulations in which the action a has been selected by tree policy, and r_a is the cumulative reward of the action a. The parameter c is the UCB constant which balances the exploration and exploitation of the action a. The first term in (7) has a higher value for actions that have been relatively more rewarding, and the second term of the equation has a higher value for actions that have a low number of exploitation compared to the total number of simulations in the tree. Tree policy selects the child node (or action) with a higher UCT value. For instance, in the selection step which Fig. 1 depicts, the node in the center is selected although it is not the most rewarding node. Because it has a higher UCT value (UCT = 25.44) compared to the most rewarding node (UCT = 24.51).

2) Expansion: MCTS executes this step if tree policy selects a simulated leaf node. Expansion step adds a new set of children nodes to the current structure of the search tree. After expansion, the tree policy selects a node among the new children nodes.

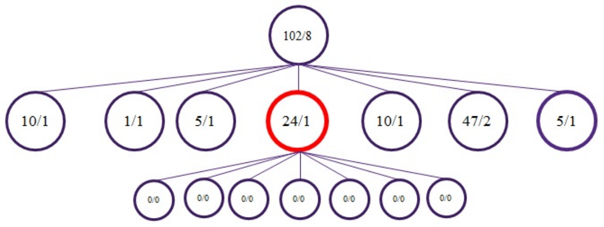

Fig. 1. An example of the selection and expansion steps in MCTS. The values shown within each node present r_a/n_a.

3) Simulation: this step executes the selected policy and determines a sequence of states using (1) to compute the cumulative reward of the selected policy, $\Re(\wp)$, by (3). Fig. 2 depicts an example of the simulation step following the selection and expansion step depicted in Fig. 1.

4) Backpropagation: as the last step of a single MCTS iteration, the algorithm backpropagates through the selected policy and updates the attributes of the nodes of tree policy. Fig. 3 presents an example of this step. To clarify, backpropagation algorithm performs the following tasks on all the nodes of $\wp_{\text{tree}}$:

- adding the computed cumulative reward, $\Re(\wp)$, to the current stored reward, r_a, for each node in tree policy
- incrementing both n_a and N by one unit.

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.

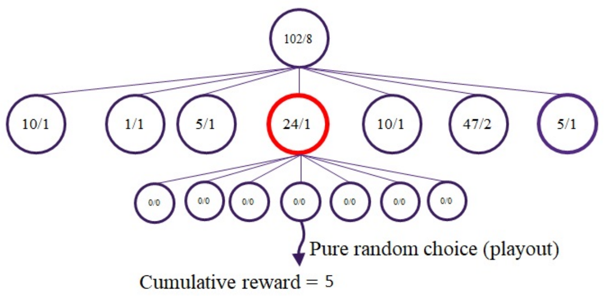

Fig. 2. Simulation step and the calculated cumulative reward.

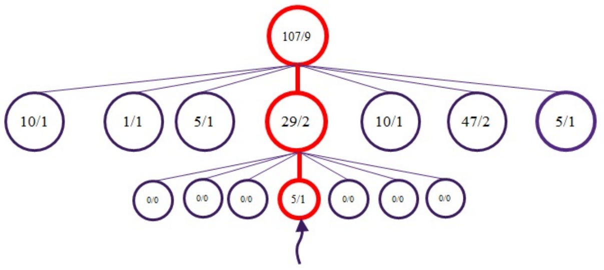

Fig. 3. Back proppagation and updating the attributes of the nodes

To summarize, in a receding horizon approach for lane change or merge, at each step of the maneuver, the MCTS executes a relatively large number of simulations over a fixed horizon choosing the actions from Table I and advancing the states according to (1). Once the termination criteria are met, it outputs the most rewarded policy to be applied. Ultimately, the automated vehicle deploys the first action (of the action sequence) and moves to the consequent state. The horizon also moves one step ahead with the agent. And continually, the tree search executes the same algorithm for the next steps of the maneuver. Algorithm 1 presents a pseudocode for the high level architecture of the MCTS algorithm. In this pseudocode, Λ denotes the tree that keeps expanding as the searching and traversing procedure continues. And η indicates a node and the other notations have been defined throughout the paper.

The presented MCTS algorithm provides the agent with a naïve strategy throughout the maneuver, where the strategy of the other agents is not concerned. Meanwhile, some of the states of each vehicle is directly influenced by the state of the surrounding vehicles (i.e. $\Gamma_{\text{collision}}(s_i, S_i')$ and $\Gamma_{\text{safe dist.}}(s_i, S_i')$). A prediction of the action sequence of other agents would benefit the host agent. Such a prediction could be integrated within the MCTS algorithm for a better outcome. For this purpose, we employ the level-k game theory, which is one of the common techniques for strategizing competitive scenarios in economy and psychology [17], [18]. And also has been recently utilized in the automated driving context in automated driving [1][2][10][11]. The following section discusses briefly the analytical and computational aspects of the level-k game theory.

IV. Integration of Level-k Game Theory

Human driver models, probabilistic techniques, and rule-based models have been developed based on analytical and

Algorithm 1 Monte Carlo tree search high level algorithm

Require: init.: $\Lambda = \eta_{\text{root}} = s_{\text{current}}$ & $\eta = \eta_{\text{root}}$

Ensure: return $\underset{a}{\operatorname{argmax}}\, \Re$

```
1:  while $N < N_{\max}$ do
2:      if $\eta =$ expanded then
3:          go to next imminent layer (children of $\eta$);
4:          $\eta =$ selection($\eta_{\text{children}}$);
5:      else
6:          if $\eta \neq$ simulated then
7:              execute the default policy selection;
8:              simulation($\eta$, $\Lambda$, $\wp$);
9:              $N = N + 1$;
10:             backpropagation($\eta$, $\Lambda$);
11:         else
12:             expansion($\eta$, $\Lambda$);
13:         end if
14:     end if
15: end while
16: $\hat{\eta} = \underset{\eta \in \Lambda_{\text{first layer}}}{\operatorname{argmax}} \Re(\wp)$;
17: map: $\hat{\eta} \Rightarrow \hat{a}$
18: return $\hat{a}$;
```

experimental methods for prediction purposes in automated driving. The complexity of mentioned methods becomes significant in the situations where the interactions between the drivers emerge. The absence of direct communication between the vehicles adds to the existing challenges. Hence, there is a need for a technique accounting for behavioral and strategizing characteristics of human drivers. Game theoretic approaches have gained attention in automated driving recently. In the application of game theory in economy and psychology, cognitive hierarchy theory describes the depth of reasoning of agents in predicting the behavior of other interacting agents in a strategic game [19][20]. In level-k game theory, an agent with a particular depth of rationalization assumes the other agent(s) possess a lower degree of sophistication. Then estimates (predicts) the course of action of the other agents accordingly. And ultimately executes its own actions with consideration of the others' predicted actions.

An agent with the k^{th} level of reasoning, assumes the competing agent(s) is (are) of one lower level which is k-1 and predicts the sequence of actions they are about to deploy over a horizon. In this assumption, the primary agent predicts what actions the other agent(s) applies considering the other agent(s) is of one lower level, that is $k-2$. And this hierarchy continues until the level of reasoning of the predicted agent reaches 0, which refers to a naïve agent that plays non-strategically. Based on experimental studies, human beings are rarely capable of depth of reasoning beyond level-2 [21].

In this study, similar to [1], a level-0 vehicle executes its MCTS under the assumption that all surrounding objects and agents are stationary. A level-1 vehicle assumes all the other vehicles are of level-0. And after predicting the applied

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.

action sequence of all the surrounding vehicles, runs a level-1 MCTS for itself with respect to the predicted states of traffic. A level-2 vehicle follows the same pattern assuming all the interacting vehicles are level-1. After the algorithm meets the termination criteria, the MCTS algorithm identifies and outputs the action with the highest reward. From the level-k theory point of view, the cumulative reward of the applied policy by the i^{th} agent of level l is given by

$$\Re_l(\wp_i) = \sum_{k=0}^{m} d^k . r_i(a_k, s_k|\hat{\wp}_{\text{others}}) \tag{8}$$

where $\hat{\wp}_{\text{others}}$ is the predicted policy of all of the surrounding vehicles by the ego vehicle.

V. Results and discussion

In the test cases presented in this section, the sampling time, Δt, is 0.25 sec. and the length of horizon is 12. The discount factor d is set to 0.8 and number of MCTS simulations were varied between 500 to 1000 to evaluate the influence of number of simulations on the performance of the algorithm. Entire algorithms, functions, and the visualization tool were programmed in C using multi-threading techniques to increase the speed of computation for mixed-level multi-agent test cases. The desired speed of vehicles was set to 22.35 m/s (= 50 mph) and the road geometry is consistent with U.S. highways.

Table II represents the computation time of a single batch of ego's MCTS execution with different levels and for different numbers of interacting vehicles. The number of iterations of each MCTS, for both ego's trajectory planning or other agents' prediction, is set to 500 for this table. Also, the algorithm was executed on a regular performance computer with the Microsoft Windows® OS, 2.6 GHz CPU and 8 GB RAM, and four threads. The reported computation time is the average of five separate executions.

TABLE II
Computation time of MCTS algorithm in millisecond

Ego's level	Number of other vehicles *1*	*2*	*3*	*4*
0	28	28	28	28
1	55	77	94	117
2	89	134	177	248

About fifty test cases were executed with different initial conditions, stationary obstacles, number of lanes, and the level of decision making vehicles. Table III represents the statistics of collision avoidance of the vehicles with different levels of reasoning. Similar to the representation in [1] and [2], the percentage shows the ratio of total number of collision avoidance events to total number of interactions between vehicles with the corresponding levels.

Fig. 4 shows a test case including multi-level agents in a multi-lane scenario. The level-0 agent performed a successful trajectory planning as it changed the lane while approaching the stationary obstacle while maintaining the desired speed. The level-1 agents performed cautious maneuvers as they consider the surrounding agents are all level-0. The level-2 agent also successfully changed its lane while interacting

TABLE III
Statistics of collision avoidance with respect to levels

Scenario	Collision avoidance stats.
level-0 & level-0	52%
level-0 & level-1	86%
level-0 & level-2	57%
level-1 & level-1	80%
level-1 & level-2	82%
level-2 & level-2	63%

with the level-1 agents. The level-2 agent predicted the motion of other agents assuming they all are level-1 which was consistent with the actual assigned level of the interacting agents. No collision occurred in this simulation. However, a collision would be probable in case the level of other agents were zero as they would assume the level-2 agent is a stationary object.

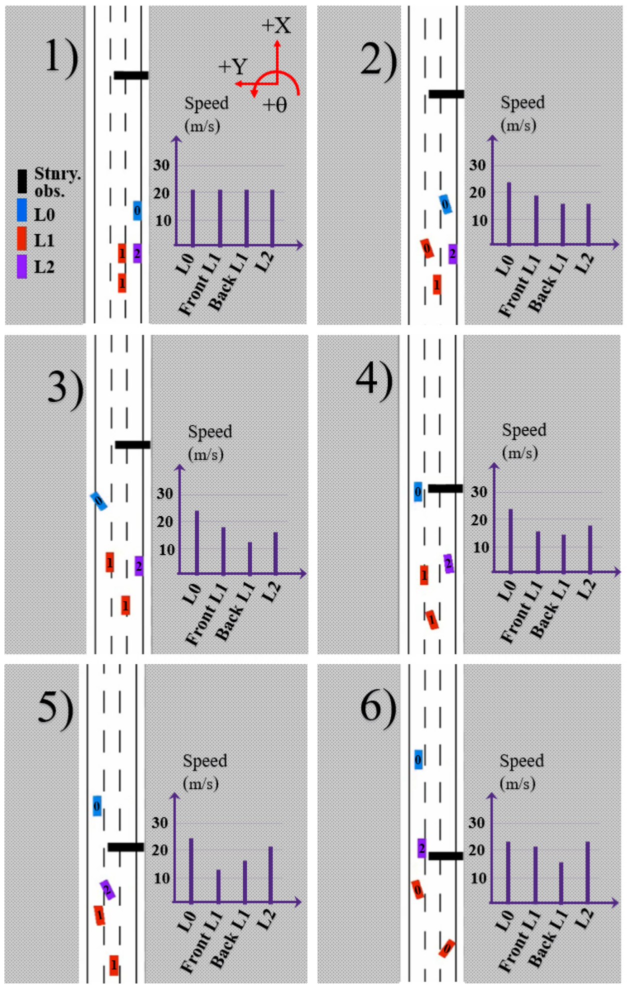

Fig. 4. Test case with different level agents in interaction. The labels of plots are in order of simulation time.

Another simulated test is presented in Fig. 5 which includes higher moving and stationary population. Even though some level-0 and level-2 agents were about to collide in some instances (part 6 of Fig. 5), no collision occurred in this case while all the agents successfully planned their motion. Also, level-1 agents mostly yield in their interactions and level-2 agents performed aggressive maneuvers as they are capable of predicting the cautious behavior of level-1 agents.

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.

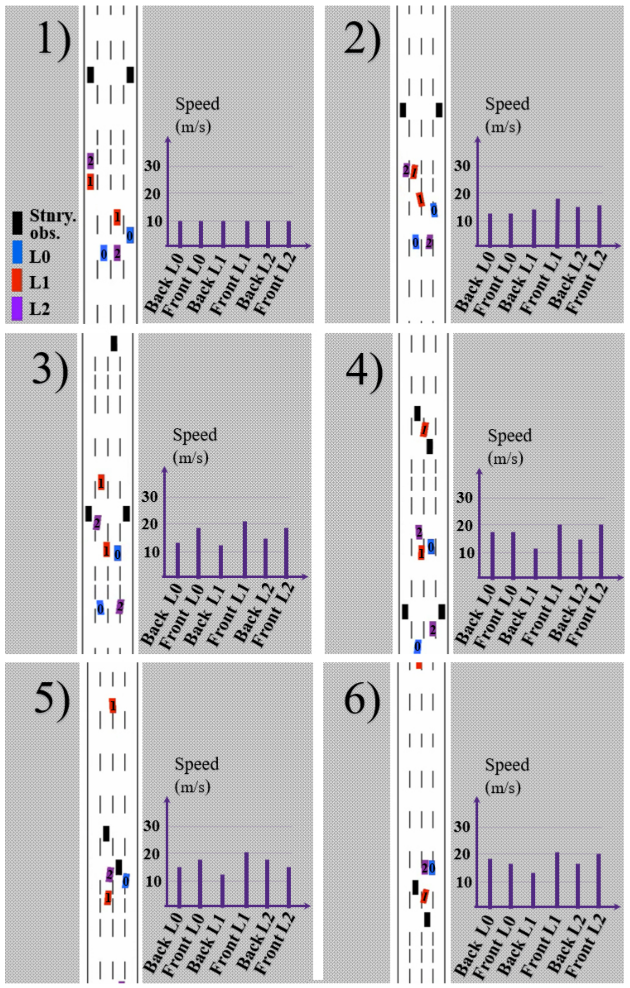

Fig. 5. Test case with higher population of interacting vehicles. The labels of plots are in order of simulation timing.

VI. Conclusion

In this study, a fusion of Monte Carlo tree search and level-k game theory was utilized in lane change maneuvers to predict the planned trajectory of the other surrounding vehicles. MCTS executes a large number of simulations over a fixed horizon and returns an action with the highest expected reward. Level-k game theory takes the strategizing level of the playing agents into account. Higher level of reasoning indicates a more sophisticated strategy in playing an interactive game. Combining level-k game theory with MCTS is expected to benefit the outcome of the entire trajectory planning mechanism while interacting with other vehicles in highway scenarios. In this research, the result of MCTS and level-k game theory in trajectory planning demonstrated real-time motion planning. Albeit, there are several components of the proposed framework that need more investigation. By definition of the level-k game theory, reasoning level of interacting agents are presumed constant and exactly one unit lower than the ego's level. Cognitive hierarchy theory (CHT), allows variety of reasoning levels among the interacting agents. In our ongoing work, we are now estimating the level of other agents within a CHT framework.

References

[1] I. Kolmanovsky, A. Girard, Y. Yildiz, and N. Li, "Game theoretic modeling of vehicle interactions at unsignalized intersections and application to autonomous vehicle control," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 3215–3220.

[2] N. Li, D. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on control systems technology*, vol. 26, no. 5, pp. 1782–1797, 2017.

[3] A. R. Dollar and A. Vahidi, "Predictive coordinated vehicle acceleration and lane selection using mixed integer programming," in *ASME 2018 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2018.

[4] B. HomChaudhuri, A. Vahidi, and P. Pisu, "Fast model predictive control-based fuel efficient control strategy for a group of connected vehicles in urban road conditions," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 760–767, 2016.

[5] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[6] A. R. Dollar and A. Vahidi, "Chance-constrained automated vehicles in hazardous merging traffic," in *9th IFAC International Symposium on Advances in Automotive Control*, Orléans, France, 2019.

[7] E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," *IFAC Proceedings Volumes*, vol. 20, no. 5, pp. 221–226, 1987.

[8] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 593–600.

[9] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, and F. Mujica, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[10] 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 *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 727–733.

[11] D. Oyler, A. G. Y. Yildiz, N. Li, and I. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 1705–1710.

[12] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo tree search: A new framework for game AI," in *AIIDE*, 2008.

[13] T. Pepels, M. Winands, and M. Lanctot, "Real-time Monte Carlo tree search in Ms Pac-Man," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 6, no. 3, pp. 245–257, 2014.

[14] A. Rimmel, O. Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai, "Current frontiers in computer Go," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 229–238, 2010.

[15] P. Ciancarini and G. P. Favini, "Monte Carlo tree search in Kriegspiel," *Artificial Intelligence*, vol. 174, no. 11, pp. 670–684, 2010.

[16] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[17] A. Arad, , and A. Rubinstein, "The 11-20 money request game: A level-k reasoning study," *American Economic Review*, vol. 102, no. 7, pp. 3561–73, 2012.

[18] D. Stahl, "Evolution of Smart$_n$ players," *Games and Economic Behavior*, vol. 5, no. 4, pp. 604–617, 1993.

[19] M. Costa-Gomes, V. P. Crawford, and B. Broseta, "Cognition and behavior in normal-form games: An experimental studys," *Econometrica*, vol. 69, no. 5, pp. 1193–1235, 2001.

[20] C. F. Camerer, T. H. Ho, and J. K. Chong, "A cognitive hierarchy model of games," *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, 2004.

[21] R. Nagel, "Unraveling in guessing games: An experimental study," *The American Economic Review*, vol. 85, no. 5, pp. 1313–1326, 1995.

Authorized licensed use limited to: Middlesex University. Downloaded on November 17,2020 at 00:29:37 UTC from IEEE Xplore. Restrictions apply.