Efficient Game-Theoretic Planning With Prediction Heuristic for Socially-Compliant Autonomous Driving

Chenran Li ⬤, Tu Trinh ⬤, Letian Wang ⬤, Changliu Liu ⬤, *Member, IEEE*, Masayoshi Tomizuka ⬤, *Life Fellow, IEEE*, and Wei Zhan ⬤, *Member, IEEE*

Abstract—Planning under social interactions with other agents is an essential problem for autonomous driving. As the actions of the autonomous vehicle in the interactions affect and are also affected by other agents, autonomous vehicles need to efficiently infer the reaction of the other agents. Most existing approaches formulate the problem as a generalized Nash equilibrium problem solved by optimization-based methods. However, they demand too much computational resource and easily fall into the local minimum due to the non-convexity. Monte Carlo Tree Search (MCTS) successfully tackles such issues in game-theoretic problems. However, as the interaction game tree grows exponentially, the general MCTS still requires a huge amount of iterations to reach the optima. In this letter, we introduce an efficient game-theoretic trajectory planning algorithm based on general MCTS by incorporating a prediction algorithm as a heuristic. On top of it, a social-compliant reward and a Bayesian inference algorithm are designed to generate diverse driving behaviors and identify the other driver's driving preference. Results demonstrate the effectiveness of the proposed framework with datasets containing naturalistic driving behavior in highly interactive scenarios.

Index Terms—Agent-based systems, autonomous agents, intelligent transportation systems.

I. INTRODUCTION

WHILE rapid progress has been achieved on autonomous driving in recent years, tackling social interactions among autonomous vehicles and human drivers in real world remains one of the greatest challenges. Under social interactions, vehicles' mutual impact on others has to be captured. Otherwise, it may lead to either excessively conservative and inefficient interactions or catastrophic accidents due to insufficient social interaction understanding.

Game-theoretic planning frameworks [1]–[4] have been widely utilized to tackle social interactions, where the influence
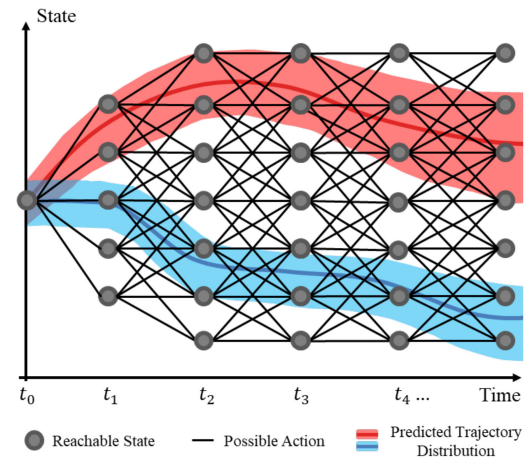
Fig. 1. Each node on the plot represents a state that can be visited by the searching algorithm, while the blue and red areas represent the suggested valuable area from the prediction models.

among agents is modeled in the system. These frameworks consider the interaction system including a robot vehicle called the ego agent and other interactive vehicles. Usually, the actions of other agents are formulated as a function of the ego agent's action, forming a closed-loop interaction system. To solve these game-theoretic interaction problems, many of the existing works [5]–[7] exploit optimization-based approaches. However, they are too computationally inefficient and would easily fall into a local minimum, especially for non-convex problems.

To address such problems, Monte Carlo Tree Search (MCTS) has been considered in driving interaction problems [8], [9]. Without the severe impact of the domain knowledge such as behavior assumptions or end conditions, MCTS can effectively evaluate the whole search space and successfully find the global optima. However, as a typical sampling method, MCTS suffers from notorious inefficiency problems. As the tree expands, though the whole space is covered and the optimal solution can be searched on a fine-grained scale, a huge amount of memory and iterations are required.

Nevertheless, our insight is that, as shown in Fig. 1, there are only a few branches that are truly valuable for searching in terms of dynamic feasibility, driving comfort/safety, interaction pattern, etc., though the tree grows rapidly. Such prior knowledge regarding which valuable areas should be searched can come from multiple sources.

We resort to the easily accessed and well established behavior prediction models [10]–[13], which can efficiently predict possible future behavior patterns. These behavior pattern predictions, though sometimes coarse or not highly accurate, can serve as an important heuristic to guide the search of MCTS, so that computational resources can be efficiently allocated at the valuable branches. However, the natural integration of the prediction heuristics into the search process is not trivial and requires delicate design. For example, overly strong prior knowledge would deprive MCTS of the capability to reach the globally optimal solution. Therefore, in this letter, we propose a novel prediction-heuristic tree search architecture to achieve efficient game-theoretic planning in interactive driving scenarios.

Moreover, in real-world interactions, human drivers may exhibit diverse social attributes. For example, in terms of driving style, while some drivers are courteous, there are also aggressive drivers [14]–[16]. To operate safely and efficiently in these social interactions, the autonomous vehicles should be able to: 1) exhibit behaviors with diverse driving styles to adapt to different situations and explicitly express its own driving intentions to others; and 2) identify others' driving styles online for conflict resolution and collision avoidance at an early stage. However, such social behavior has not been captured in previous MCTS-based driving methods, in which all the agents are modeled with the same fixed policy. Such modeling is lacking in the ability to generate diverse driving styles and recognize others' driving styles, resulting in severe safety or efficiency issues in intensive social interactions. In this letter, we design a socially-compliant reward capturing the selfish and courteous driving preference, by which driving behavior with diverse styles could be generated. A Bayesian inference algorithm is utilized to identify other drivers' driving styles online by estimating the parameter in the socially-compliant reward.

The key contributions of this work are as follows.
- Proposing a new framework by incorporating prediction heuristic into MCTS architecture for efficient game-theoretic planning in interaction-aware driving scenarios.
- Designing a social-compliant reward and a Bayesian inference algorithm within the MCTS architecture, to generate diverse driving behaviors and identify others' driving styles online.
- Conducting comprehensive experiments with naturalistic data, which demonstrates the effectiveness of prediction heuristic and the game-theoretic interaction in highly interactive driving scenarios.

To the best of our knowledge, this letter is the first to incorporate prediction heuristic and socially-compliant reward into the MCTS framework for socially interactive driving tasks.

II. PROBLEM FORMULATION

In this work, we consider the interaction system with two agents including a robot vehicle called the ego agent and an interactive vehicle called the opponent agent. The superscript E denotes the ego agent and O denotes the opponent agent. The interaction system satisfies an overall dynamics:

$$\mathbf{x}_{t+1} = f\left(\mathbf{x}_t, u_t^E, u_t^O\right), \tag{1}$$

where $\mathbf{x}_t = \left\{\mathbf{x}_t^E, \mathbf{x}_t^O\right\}$ is the joint state of the ego agent and the opponent agent at time step t representing their dynamic states such as position, velocity and etc. u_t is the control input at time step t representing the action of each agent such as acceleration,
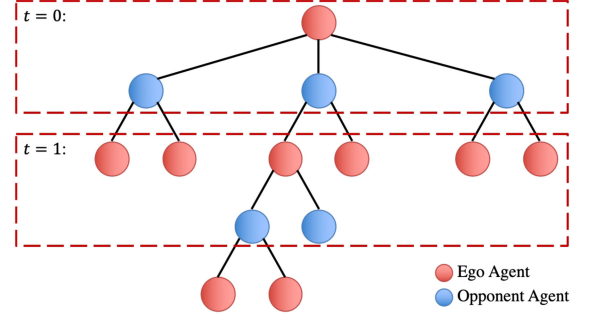

Fig. 2. Game Tree Representation of Interaction.

steering angle and etc. Let N be the horizon of the interaction. The action sequence of the each vehicle in the next N steps are denoted as $U_N^{E/O} = \left[u_0^{E/O}, \ldots, u_{N-1}^{E/O}\right]^T$.

The agents in the interaction are optimizers that their control inputs maximize the cumulative reward function of each own noted by R^E and R^O. Thus, the optimal action sequence U_N^* that each vehicle will take can be represented as:

$$U_N^{E*} = \arg\max_{U_N^E} R^E\left(\mathbf{x}_0, U_N^E, U_N^{O*}\right),$$

$$U_N^{O*} = \arg\max_{U_N^O} R^O\left(\mathbf{x}_0, U_N^{E*}, U_N^O\right). \tag{2}$$

To better explain the diverse driving preferences of the opponent agent and generate socially-compliant behavior for the ego agent, the courtesy factor should be considered in the reward function during the planning. Similar to [5], [17], the reward for modeling the agents in the proposed method is designed as a linear combination of the egoism reward term and the courtesy reward term. For instance, the ego agent's reward is formulated as:

$$R^E\left(\mathbf{x}_0, U_N^E, U_N^O\right) = \gamma R_{\text{Egoism}}^E\left(\mathbf{x}_0, U_N^E, U_N^O\right)$$
$$+ (1 - \gamma) R_{\text{Courtesy}}^E\left(\mathbf{x}_0, U_N^E, U_N^O\right), \tag{3}$$

where R_{Egoism}^E and R_{Courtesy}^E are the egoism reward term and the courtesy reward term respectively, a detailed definition of which can be found in Section IV-A. $\gamma \in [0, 1]$ balances the reward features.

As in [6], [15], [16], we further assume that two agents are running a Stackelberg game where one agent is the leader and the other agent is a follower. Note that the Stackelberg game assumption is reasonable for most interaction scenarios since different agents always have different right-of-way in interaction. Thus we can always select the vehicle with a higher right-of-way as the leader. Without losing generality, in this letter we assume the ego agent as the leader agent.

Therefore, the optimization in (2) can be reformed as:

$$U_N^{E*} = \arg\max_{U_N^E} R^E\left(\mathbf{x}_0, U_N^E, U_N^{O*}\left(U_N^E\right)\right),$$

$$U_N^{O*}\left(U_N^E\right) = \arg\max_{U_N^O} R^O\left(\mathbf{x}_0, U_N^E, U_N^O\right). \tag{4}$$

To solve the optimization problem, we regard the optimal action sequence as the optimal node path in the game tree shown in Fig. 2, where the two agents make decisions sequentially at

---

Algorithm 1: Prediction Heuristic Search Architecture.

Input: Current state $\mathbf{x}_0$, K predicted trajectories
$\hat{\mathbf{Y}}_N^i = \{\hat{\mathbf{y}}_1^i, \ldots, \hat{\mathbf{y}}_N^i\}$, $i = 1, \ldots, K$.

Output: Root node n_0^E of the built tree

1:   Initialize $G = \{G_1^1, \ldots G_N^K\}$ from $\hat{\mathbf{Y}}_N^i$, for
    $i = 1, \ldots, K$ and $t = 1, \ldots, N$.
2:   Initialize the root node $n_0^E = (\mathbf{x}_0, \emptyset, \emptyset)$.
3:   Initialize iteration counter $Iter = 0$.
4:   **while** $Iter <$ Maximum Iterations **do**
5:     $n_l \leftarrow SELECTION n_0^E$
6:     **if** n_l is non-terminal **then**
7:       $n_N, U_N^E, U_N^O \leftarrow$ Roll-out(n_l)
8:     **else**
9:       $n_N, U_N^E, U_N^O \leftarrow n_l, U_N^E(n_l), U_N^O(n_l)$
10:    **end if**
11:    **if** $I_{unsafe}(n_N)$ **then**
12:      $q^E, q^O \leftarrow 0, 0$
13:    **else**
14:      $q^E, q^O \leftarrow R^E\left(\mathbf{x}_0, U_N^E, U_N^O\right), R^O\left(\mathbf{x}_0, U_N^E, U_N^O\right)$
15:    **end if**
16:    **while** n_l is not NULL **do**
17:      $C(n_l) \leftarrow C(n_l) + 1$
18:      $w_{conf} \leftarrow 1$
19:      **if** n_l is a ego node **then**
20:        $w_{conf} \leftarrow w_{conf}(n_l, G)$
21:      **end if**
22:      $q_s^E, q_s^O \leftarrow w_{conf} \times q^E, w_{conf} \times q^O$
23:      $Q^E(n_l), Q_s^E(n_l) \leftarrow Q^E(n_l) + q^E, Q_s^E(n_l) + q_s^E$
24:      $Q^O(n_l), Q_s^O(n_l) \leftarrow Q^O(n_l) + q^O, Q_s^O(n_l) + q_s^O$
25:      $n_l \leftarrow$ parent of n_l
26:    **end while**
27: **end while**
28: **return** n_0^E

Algorithm 2: Function for Algorithm 1.

1: **function** SELECTION n_t
2:     **while** n_t is non-terminal **do**
3:       **if** n_t is not fully expanded **then**
4:         $n_l \leftarrow$ expand n_t with an untried action.
5:         **while** $I_{unsafe}(n_l)$ **do**
6:           $n_l \leftarrow$ expand n_t with an untried action.
7:         **end while**
8:         **return** n_l
9:       **else**
10:        $n_l \leftarrow$ solving (5) with Q_s^E when n_t is ego node;
          Q_s^O when n_t is opponent node.
11:       **end if**
12:     **end while**
13: **end Function**
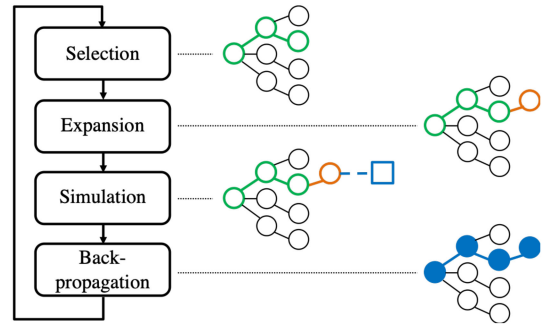
Fig. 3.   The Structure of The General MCTS Algorithm.

each search step. Formally, with the discretized action space, at search iteration t, we have an ego node $n_t^E = \left(\mathbf{x}_t, U_t^E, U_t^O\right)$ and an opponent node $n_t^O = \left(\mathbf{x}_t, U_{t+1}^E, U_t^O\right)$. The root node and the terminal node can be represented as $(\mathbf{x}_0, \emptyset, \emptyset)$ and $\left(\mathbf{x}_N, U_N^E, U_N^O\right)$ respectively.

When searching for optimal action sequences in such game tree problems, Monte Carlo Tree Search (MCTS) has achieved significant success in various applications, especially in multi-agent setting [18]–[20]. However, the search space grows exponentially as the action space and the time space extend, which leads to low efficiency and infeasible practical deployment due to computation limitations. Therefore, in this letter, we propose the prediction-heuristic search architecture based on general MCTS for efficient searching and planning.

III. PREDICTION HEURISTIC SEARCH ARCHITECTURE

As shown in Fig. 3, the general MCTS algorithm includes mainly four steps per search iteration that are selection, expansion, simulation, and back-propagation [21]. At the selection phase, the algorithm will recursively select the most valuable child node until reaching the leaf node. After selecting the node, the algorithm will expand the tree at the selected node according to the available actions. During the simulation phase, the algorithm will use a roll-out policy to simulate the game from the expanded node to the end. At the back-propagated phase, the algorithm will extract the rewards from the roll-out sample and use them to update the average rewards through the selected nodes. After reaching the defined maximum iterations, the solution can be extracted by selecting the maximum rewards child node for each layer.

As the MCTS gradually evaluates the whole interaction action space, the game tree grows exponentially, which would result in either failure in finding optimal solutions or computationally-infeasible in practical deployment. However, in terms of dynamics feasibility, driving comfort/safety, and interaction pattern, there are in fact only a few branches that are truly meaningful to search. Thus, our idea here is to use the predicted future trajectory, which is easily accessed from well-developed prediction models, to guide the exploration toward fruitful branches of the tree. The pseudo-code of the proposed method is presented in Algorithm 1, 2, where the variables used in the algorithm are summarized in the Table I and will be defined in detail in the following subsections.

Similar to the general MCTS, let $C(n_t)$ denote the number of times that the input node has been visited. $Q^E(n_t), Q^O(n_t)$ denotes the total reward of all roll-out trajectories that pass the node n_t for the ego agent and the opponent agent respectively. To keep the reward unbiased when extracting the planned trajectory, we introduce the searching rewards. Let $Q_s^E(n_t), Q_s^O(n_t)$ denote the total searching reward of all roll-out trajectory that pass the node n_t for ego agent and opponent agent respectively. The

TABLE I
NOTATIONS IN ALGORITHM 1

Notation	Description
$\mathbf{x}_0$	Current state of the interaction system
$\hat{\mathbf{Y}}_N^i$	i^{th} prediction trajectory with N steps
$\hat{\mathbf{y}}_t^i$	Predicted state in $\hat{\mathbf{Y}}_N^i$ at step t
G_t^i	Confident range introduced by $\hat{\mathbf{y}}_t^i$
G	Set of all confident ranges
w_{conf}	Confident weight
$U_N^{E/O}(n)$	Action sequence stored in terminal node
$I_{unsafe}(n)$	Indicator function of safety constraints
$Q^{E/O}(n)$	Total reward of all roll-out trajectories
$Q_s^{E/O}(n)$	Total searching reward of all roll-out trajectories
$q_s^{E/O}$	Reward of a roll-out trajectory
$q^{E/O}$	Searching reward of a roll-out trajectory

selection policy is defined as

$$\arg\max \frac{Q_s^{E/O}(n_l)}{C(n_l)} + c\sqrt{\frac{2\ln C(n_t)}{C(n_l)}}.$$

$$s.t. n_l \in \text{children of } n_t \qquad (5)$$

The first term in the equation is the average reward of passing node n_l to encourage the exploitation of high-reward actions. The second term is an exploration bonus to encourage the exploration of less-visited actions, where c is a constant parameter to balance the trade-off between exploitation and exploration [20].

A. Prediction Heuristic Exploration

Within the MCTS process, searching is conducted for the ego agent and opponent agent iteratively as in Fig. 2. The searching nodes for the opponent agent are actually embedding a prediction module inside the planner to simulate the whole interaction. Though precise and explainable solutions can be searched, the search space is evaluated uniformly. Due to the fact that the possible interaction and driving patterns of the two vehicles are usually limited, the computation resources are mostly spared in uninformed search space, leading to low efficiency. On the other hand, nowadays many efficient learning-based prediction methods have been developed [10]–[12]. With actual driver interaction captured in real human data, the predicted trajectory from these learned prediction models usually presents feasible interaction patterns of the two agents, like one yielding car and one passing car. Though predicted trajectories may not be as precise and accurate as that in the tree, they are still highly valuable prior knowledge that provides a good heuristic for the search. To integrate such heuristics into the search process while keeping sufficient exploration, we design a confidence range around the prediction trajectory and the associated confidence weight to guide exploration.

1) Confidence Range: From the prediction model, we can retrieve K possible future trajectories and the corresponding probabilities for the opponent agent, which may represent one interaction or driving pattern. Let $\hat{\mathbf{Y}}_N^i = \{\hat{\mathbf{y}}_1^i, \dots, \hat{\mathbf{y}}_N^i\}$ and p_i denote the ith predicted trajectory and its probability of being executed respectively. Therefore, the state at time step t can be modeled as a random distribution with mean $\hat{\mathbf{y}}_t^i$ and variances Σ_t^i, where the variances Σ_t^i in the predicted trajectory can be calculated by methods discussed in [22], [23]. Thus, each predicted trajectory introduces a confidence range G_t^i at each

time step by Mahalanobis distance, which is given as

$$G_t^i = \left\{ \mathbf{x}_t^O \mid \left(\mathbf{x}_t^O - \hat{\mathbf{y}}_t^i\right)^T \Sigma_t^{i-1} \left(\mathbf{x}_t^O - \hat{\mathbf{y}}_t^i\right) \leq \rho \right\}, \qquad (6)$$

where ρ is a constant parameter that controls the size of G_t^i. Note that after choosing parameter ρ, the size of G_k^i only depends on the variance of the prediction trajectory. When the predicted trajectory has a large variance which means large uncertainty exists in the prediction heuristic, the size of G_k^i will also increase, encouraging exploration in a larger space.

2) Confidence Weight: With the confidence range, we introduce the confidence weight to balance the trade-off between maximizing reward and following the prediction. When back-propagating the reward to each layer, the searching reward of the roll-out sample $q_s^{E/O}$ will be weighted by its confidence weight w_{conf} as in line 18 in Algorithm 1. To encourage wider exploration, the confidence weight should remain uniform in the whole confidence range instead of focusing on the mean. Branches inside multiple confidence ranges should also raise more explore attention. Based on these insights, w_{conf} is then defined as the summation of associated probabilities p_i of all confidence ranges that node n_t is inside, namely

$$w_{\text{conf}}(n_t, G) = \sum_{i \in V} p_i,$$

$$V = \left\{ i \mid \mathbf{x}_t^O(n_t) \in G_t^i, G_t^i \in G \right\}, \qquad (7)$$

where G is the set of all confidence ranges. By multiplying the reward with the confidence weight, the algorithm will explore more those branches that are more interaction-feasible in practice and have a higher reward.

B. Roll-Out Policy

In the simulation phase, the algorithm will use a roll-out policy to simulate the game from the expanded node to the end. In this process, efficiently generating valuable sample trajectories is important for the effective evaluation of the expanded nodes. Our insight is that the prediction heuristic introduced earlier in the expansion phase can also serve as valuable prior knowledge in this simulation phase. Therefore, we also exploit the prediction heuristic in the roll-out policy. Specifically, when generating a roll-out trajectory from an expanded node, the algorithm will check the confidence range and related prediction trajectory that the node belongs to. Then the roll-out trajectory will be generated within confidence ranges introduced by the same prediction trajectory, by a random action selection process. Meanwhile, in terms of driving comfort, the roll-out action is constrained by comfort jerk j_{comf}, namely

$$|a_{t+1} - a_t| \leq j_{\text{comf}}, \qquad (8)$$

where a_t is the corresponding acceleration of the action u_t.

C. Safety Constraints

During searching, identifying unsafe nodes is not only important for the feasibility of the result but also for increasing the searching efficiency. Let $I_{unsafe}(n_t)$ be the indicator function of safety constraints that will be true when the trajectory between the root node and input node violates the given safety constraints such as collision with other agents and road edge. There are two procedures in which the safety constraints will be evaluated.

Algorithm 3: Reward Parameter Bayesian Inference.

Input: M reward candidate parameter samples γ^i,
corresponding weights ω^i_{k-r-1}, and observations $\hat{\mathbf{x}}_{k-r:k}$
Output: Updated weights ω_{k-r}, estimated parameter
γ_{k-r}.
1: **if** k-r-1=0 **then**
2:     Initialize candidate reward parameter samples γ^i and
         corresponding weights ω_{k-r-1}
3: **end if**
4: **for** all M candidate reward parameter samples **do**
5:     Update $\omega^i_{k-r} \leftarrow \omega^i_{k-r-1} \times p\left(\hat{\mathbf{x}}_{k-r:k}|\gamma^i\right)$, (12)
6: **end for**
7: Normalize $\omega_{k-r} \leftarrow \omega_{k-r}/\sum_{i=1}^{M}\omega^i_{k-r}$
8: Compute $\gamma_{k-r} \leftarrow \sum_{i=1}^{M}\gamma^i\omega^i_{k-r}$

---

1) Sample Evaluation: As shown at line 11 in Algorithm 1, the safety constraints will be evaluated before calculating the reward of the roll-out sample. If the sample trajectory is unsafe, the reward of it will be fixed to zero, which penalizes the average reward of the branch for safety but save the branch for remaining potential trajectories.

2) Expanding Tree: After expanding nodes, it is necessary to check if they satisfy the safety constraints, as at line 5 in Algorithm 2. If the indicator function gives that the expanded node is unsafe, all the succeeding branches after it will be blocked by this unsafe node. Therefore, the algorithm will remove the unsafe node when expanding the tree to reduce the searching space.

IV. SOCIALLY-COMPATIBLE DRIVING

Safe and efficient interaction necessitates the inclusion of diverse social preferences in driving. Thus in this section, we elaborate on the socially-compatible reward design and the utilized Bayesian inference algorithm to online estimate the reward parameter.

A. Socially-Compatible Reward Design

As introduced in (3), for one selected ego vehicle, the socially-compatible reward is designed as the linear combination of egoism reward term $R^E_{\text{Egoism}}\left(\mathbf{x}_0, U^E_N, U^O_N\right)$ and the courtesy reward term $R^E_{\text{Courtesy}}\left(\mathbf{x}_0, U^E_N, U^O_N\right)$. The parameter γ balances the two terms, which can be utilized to generate behaviors with different styles as shown in Section V-B. The egoism reward is defined to capture an agent's own utilities:

$$R^E_{Egoism}\left(\mathbf{x}_0, U^E_N, U^O_N\right) = \theta^T \sum_{t=0}^{N-1} \phi\left(\mathbf{x}_t, U^E_t, U^O_t\right), \quad (9)$$

where $\phi \in \mathrm{R}^3$ denotes the three egoism utility feature: efficiency, comfort, and safety. $\theta \in \mathrm{R}^3$ denotes their relative weights, which can be manually tuned or learned via Inverse Reinforcement Learning [17], [24]. Inspired by [14], the courtesy reward term can be represented by the opponent agent's egoism reward, that is

$$R^E_{\text{Courtesy}}\left(\mathbf{x}_0, U^E_N, U^O_N\right) = R^O_{\text{Egoism}}\left(\mathbf{x}_0, U^E_N, U^O_N\right). \quad (10)$$

B. Reward Parameter Estimation

Human behaviors are naturally heterogeneous, time-varying, and stochastic [25], [26]. Driving preferences may vary in individuals and circumstances. Hence, we should infer the opponent agent's current driving policy by estimating the reward parameter γ in (3).

The insight of reward update is that, though human drivers could not communicate directly, their historic behaviors could leak their mental state, which can be leveraged to infer their driving preferences. Technically, based on maximum entropy, the probability of one candidate reward parameter γ is proportional to the probability of the observed trajectory $\hat{\mathbf{x}}_{k-r:k}$ conditional on that candidate reward parameter:

$$p\left(\gamma|\hat{\mathbf{x}}_{k-r:k}\right) \propto p\left(\hat{\mathbf{x}}_{k-r:k}|\gamma\right)p(\gamma), \quad (11)$$

where we look r steps to the past from current time step k, and $p(\gamma)$ is the prior probability. The probability of the observed trajectory under one candidate reward parameter is proportional to its reward under that candidate reward parameter:

$$p(\hat{\mathbf{x}}_{k-r:k}|\gamma) \approx \frac{e^{R(\hat{U}_{k-r:k},\gamma)}}{\sum_{U \in \mathbf{U}} e^{R(U_{k-r:k},\gamma)}}, \quad (12)$$

where $\mathbf{U}$ is the set of all possible actions, and $\hat{U}$ is the action resulting in the observed trajectory. $R(U_{k-r:k},\gamma)$ denotes the reward gained under action $U_{k-r:k}$ and parameter γ, as detailed in (3).

With (11) and (12), we formulate a Bayesian inference algorithm to estimate the reward parameter for one agent on the fly. The algorithm is outlined in Algorithm 3. Line 2 samples M reward parameters γ^i and initializes corresponding weights $\omega^i(\gamma^i)$. Line 5 updates weights for each sample. Line 8 estimates reward parameter γ_{k-r} as the mean of the posterior distribution, which is taken as the reward parameter in the current time step γ_k by the assumption of a small change of reward parameter γ over the observed horizon N.

V. EXPERIMENTS

In this section, a series of experiments were conducted to test the performance of the proposed algorithm. In Section V-A, we analyze the optimality and efficiency improvement brought by the proposed algorithm in real human interaction scenarios. The detailed searching information is provided showing the effectiveness. In Section V-B, we deploy the planner in several closed-loop simulations with different behaviors generated by the designed socially-compatible reward. In Section VI, we further discuss the implementation usage of the proposed framework.

All experiments were generated from real-world driving scenarios from the INTERACTION Dataset [27]. There are different kinds of scenarios in the INTERACTION Dataset, and we utilized the roundabout scenario for our experiments. The insight behind such a choice is that interaction essentially happens between agents that are driving toward the same region and thus have a conflict with each other [28]. In such a general definition, interaction can be modeled regardless of scenarios. Consequently, our studies and conclusions can also apply to other driving scenarios, like intersections, merging scenarios, etc. Let l_{ref} denote the length of reference path, v_{max} denote the maximum legal speed. The states of an agent were represented with the longitudinal position $s_t \in [0, l_{\text{ref}}]$ and longitudinal speed
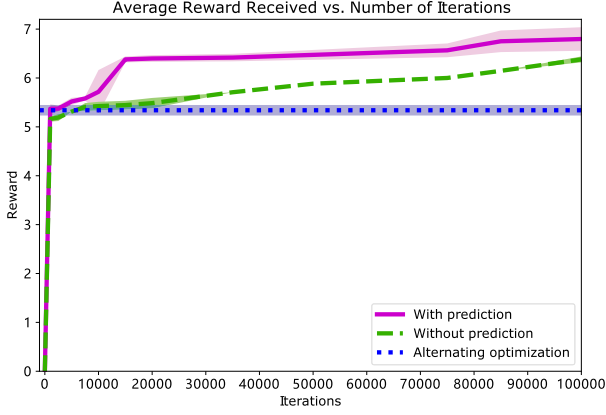
Fig. 4. Average Reward of The Output Trajectory with Different Maximum Iterations.

$v_t \in [0, v_{\max}]$ along the reference path. The agent's control input was represented as longitudinal acceleration a_t that was discretized into a list with choices as $[-3, -2, -1, 0, 1, 2]$ m/s^2. The planning time step was 0.5 s and the planning horizon $N = 5$. The reward function in these experiments was given by

$$R_{Egoism} = \theta^T \sum_{t=0}^{N-1} \left[e^{-\alpha a_t^2}, 1 - e^{-\beta v_{t+1}^2} \right]^T, \qquad (13)$$

where α, β are constant parameters. The experiments were mainly simulating around the interaction point which is defined as the cross point of the reference paths of the ego agent and opponent agent.

A. Searching Performance

In this subsection, we randomly selected a time frame before any agent reach the interaction point from the dataset, and planned the future trajectory for ten times with different random seeds by different algorithms. To focus our study on the effect of the prediction heuristic, the predicted trajectory was generated by adding the ground truth trajectory with Gaussian noise $N(0, 0.4^2)$ meter for each trajectory point.

1) Baselines: To evaluate the effectiveness of the proposed algorithm, we first implemented a baseline MCTS algorithm without the prediction heuristic, which is used in [8]. Additionally, we also implemented the alternative optimization algorithm based on the iterative best response optimization algorithm [14]. The alternating optimization algorithm will initialize each agent with a random action sequence and iteratively search for the best action sequence of each agent separately until both agents have converged.

2) Performance Comparison: As shown in Fig. 4, we first present the reward of the planned trajectory generated by the three different methods, as the search iteration increases. Note that the alternating optimization algorithm iteratively searches in two trees until both trees converge (one tree for one agent), and the other two methods only search in one tree where both two agents are nested. Consequently, one iteration in the alternating optimization algorithm represents different computation costs compared to the other two methods. For a fair comparison, we only draw the reward of the final trajectory generated by the alternating optimization algorithm. As in Fig. 4, the two

methods that search within one tree result in higher rewards than the alternating optimization algorithm. Also, benefit from the prediction heuristic, our proposed method shown with the purple curve converges faster than the general MCTS algorithm shown with the green curve, demonstrating the efficiency improvement of our method.

3) Searching Analysis: To further investigate the effect of prediction heuristic in detail, we also compared the visiting times of the maximum reward node V_{Max} and other nodes V_{Other} in the same layer and maximum depth that the algorithm was able to get. Table II presents the result with 30000 iterations. Following the formulation in the previous sections, the odd layer contained nodes reached by the ego agent's actions while the even layer contained nodes reached by the opponent agent's actions.

As shown in the Table II, both the general MCTS and the proposed algorithm visited more times on the optimal node they find. As the searching layer gets deeper, the distribution of visiting nodes becomes more uniform for the general MCTS. However, with the prediction heuristic, the proposed algorithm was able to keep the focus on the optimal branches, especially in the layer of the opponent agent's actions. Moreover, as a benefit, the proposed algorithm was also able to search deeper and provide a longer plan than the general MCTS. Meanwhile, as it maintained more samples on the optimal node, it was more convincing on the optimality of the found trajectory.

B. Closed-Loop Simulation

We conducted several closed-loop simulations to investigate the behaviors with different sets of the reward and evaluate the effectiveness of the reward parameter estimation. To generate a reactive game, both the ego agent and the opponent were planned by the proposed algorithm. The opponent agent was assumed to have knowledge of the ego vehicle's behavior parameter. The ego vehicle, however, had to figure out the opponent agent's setting with the proposed reward estimation. The simulation time step was 0.1 s. To focus our study on the proposed method, the predicted trajectory was generated by the ground truth with Gaussian noise $N(0, 0.4^2)$ meter at each trajectory point. Fig. 5 shows three simulations conducted with different reward settings.

1) Behavior Diversity: In the first simulation, the ego agent was set to be a courteous driver while the opponent agent was set to be an egoistic driver. Even though the ego agent was the leader in the game, the algorithm chose to yield the opponent agent since it received more rewards from the courtesy term.

We then switched the behavior parameter of the ego agent and opponent agent to create the second simulation. As expected, the opponent agent behaved courteously and decided to yield the ego agent even with a higher velocity.

To further study the effect of the behavior parameter, in the third experiment, both agents were set as egoistic drivers, but the ego agent was more egoistic than the opponent. As shown in the speed curves, compared with the courteous case, at the time step of 2 s, the opponent refused to decrease its speed initially. Meanwhile, for the ego agent, after realizing the opponent's behavior preference, it decided to insist on its right-of-way by increasing its speed for a little after the time step of 1.7 s. After the opponent eventually decided to yield at the time step of 2.6 s, the ego agent was able to drive with a higher speed in safe.

2) Behavior Parameter Estimation: As shown with the (e) column in Fig. 5, the algorithm was able to accurately estimate

TABLE II
VISITING TIMES FOR DIFFERENT DEPTH IN SEARCH TREE

Depth		Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9
General MCTS	V_{Max}	5077	853	142	24	4	1	0	0	0
	V_{Other}	4984	844	142	23	4	0	0	0	0
Proposed Method	V_{Max}	5259	3036	547	464	87	49	8	2	1
	V_{Other}	4948	444	497	16	75	7	8	1	0
Ratio of V_{Max} Between Methods		1.04	3.55	3.85	19.33	21.75	49.00	Inf.	Inf.	Inf.

(a) Behavior Paramters   (b) Closed-loop Trajectory   (c) Distance to Interaction Point   (d) Speed Curve   (e) Estimatied Behavior Parameters
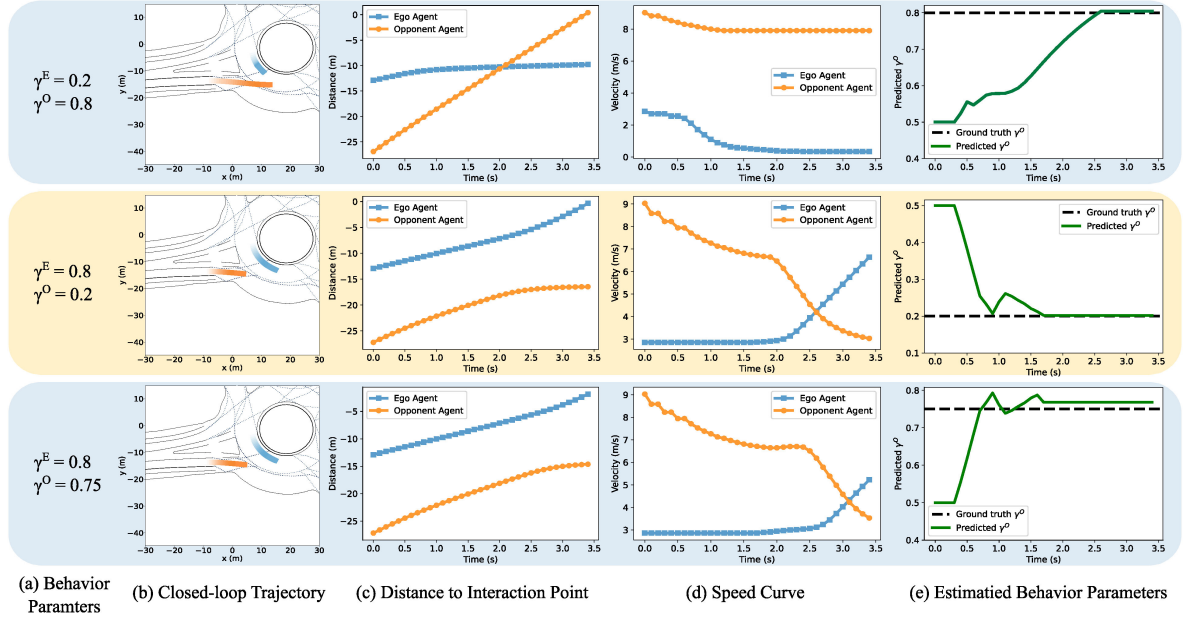
Fig. 5.   Closed-loop Simulation with Different Driving Behaviors. In (b), cars at earlier time steps are drawn more transparently for time-lapse display.

the behavior parameter of the opponent agent for all simulations. In the last two simulations, the estimation was able to reach the ground truth in 1 s, namely, 6 time steps after receiving enough observations. However, in the first simulation, the convergence speed was relatively slow. The reason for it is mainly from the ego agent's behavior. As the ego agent chose to yield and stopped at an early stage, there was not much interaction that affect the opponent's actions, which made all ω^i update more slowly than in other cases.

VI. DISCUSSION

In this section, we discuss the extension of the proposed algorithm for general scenarios and several implementation usages of the proposed algorithm in addition to decision and planning.

A. Multi-Agents Interactions

The extension of the proposed formulation and solution to multi-agent settings can be defined in *theory* without heavy effort. Following the same assumption of the Stackelberg game, the M agents can be sorted by their right-of-way. Then, each interaction time step in the game tree shown in Fig. 2 can be extended to include M layers by the sorted order. Each layer represents the possible choices of actions for one agent at one time step. The proposed prediction heuristic can be applied similarly for each predicted agent.

Though the extension takes little effort in the formulation, the multi-agent setting is still a challenging problem in practice, as the computational resources are required exponentially with

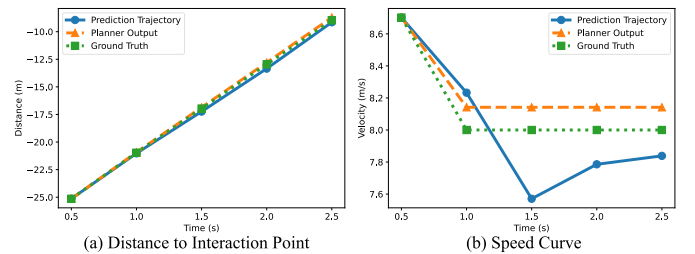(a) Distance to Interaction Point   (b) Speed Curve

Fig. 6.   Comparison of the inferred trajectory and its corresponding prediction trajectory.

the number of agents. Due to these limitations, the multi-agent problem is still hard to solve with issues of efficiency, convergence, and local optima. We believe this is an important and interesting future work that we will look into in future studies.

B. Practical Usage of the Proposed Method

During the closed-loop simulation, we also compared the inferred opponent's trajectory with its ground truth. As shown in Fig. 6, the inferred trajectory is able to recover the ground truth trajectory from its prediction with noises. Therefore, the proposed algorithm can also be used as an interaction-aware predictor to facilitate the learned model to generate better joint predictions.

VII. Conclusion

In this letter, we proposed a game-theoretic planning algorithm by incorporating the prediction heuristic to make the search much more computationally efficient. A social-compliant reward and a Bayesian inference algorithm built on it were designed to generate diverse driving behaviors and identify others' driving preferences. As demonstrated in the results, the proposed planning framework was able to generate different behaviors by the designed reward and plan more efficiently than the general MCTS algorithm. The proposed Bayesian inference algorithm can successfully infer others' driving styles from the observations.

Acknowledgment

The authors would like to thank Ran Tian for insightful advice and help.

References

[1] S. M. LaValle, "Robot motion planning: A game-theoretic foundation," *Algorithmica*, vol. 26, no. 3, pp. 430–465, 2000.

[2] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1313–1325, Aug. 2021.

[3] A. Dreves and M. Gerdts, "A generalized nash equilibrium approach for optimal control problems of autonomous cars," *Optimal Control Appl. Methods*, vol. 39, no. 1, pp. 326–342, 2018.

[4] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.

[5] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proc. Robot.: Sci. Syst.*, Ann Arbor, Michigan, 2016, pp. 1–9.

[6] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 66–73.

[7] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 9590–9596.

[8] L. Sun, M. Cai, W. Zhan, and M. Tomizuka, "A game-theoretic strategy-aware interaction algorithm with validation on real traffic data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11038–11044.

[9] K. Kurzer, C. Zhou, and J. M. Zöllner, "Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 529–536.

[10] L. Wang, Y. Hu, L. Sun, W. Zhan, M. Tomizuka, and C. Liu, "Transferable and adaptable driving behavior prediction," 2022, *arXiv:2202.05140*.

[11] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland, 2020, pp. 683–700.

[12] L. Wang, Y. Hu, L. Sun, W. Zhan, M. Tomizuka, and C. Liu, "Hierarchical adaptable and transferable networks (HATN) for driving behavior prediction," 2021, *arXiv:2111.00788*.

[13] P. Tripicchio and S. D'Avella, "Modeling multiple vehicle interaction constraints for behavior prediction of vehicles on highways," *Comput. Elect. Eng.*, vol. 98, 2022, Art. no. 107700.

[14] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proc. Nat. Acad. Sci.*, vol. 116, no. 50, pp. 24972–24978, 2019.

[15] L. Wang, L. Sun, M. Tomizuka, and W. Zhan, "Socially-compatible behavior design of autonomous vehicles with verification on real human data," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3421–3428, Apr. 2021.

[16] L. Sun, W. Zhan, M. Tomizuka, and A. D. Dragan, "Courteous autonomous cars," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 663–670.

[17] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. 29th Int. Conf. Int. Conf. Mach. Learn.*, 2012, pp. 475–482.

[18] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Decentralised Monte Carlo tree search for active perception," in *Algorithmic Foundations of Robotics XII*. Cham, Switzerland: Springer, 2020, pp. 864–879.

[19] B. Kartal, E. Nunes, J. Godoy, and M. Gini, "Monte carlo tree search for multi-robot task allocation," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, pp. 4222–4223.

[20] C. B. Browne et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[21] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo tree search: A new framework for game ai," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2008, vol. 4, no. 1, pp. 216–217.

[22] L. Sun, X. Jia, and A. Dragan, "On complementing end-to-end human behavior predictors with planning," in *Proc. Robot.: Sci. Syst.*, Jul. 2021, doi: 10.15607/RSS.2021.XVII.037.

[23] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*. Chicago, IL, USA, 2008, vol. 8, pp. 1433–1438.

[25] A. Abuduweili and C. Liu, "Robust nonlinear adaptation algorithms for multitask prediction networks," *Int. J. Adaptive Control Signal Process.*, vol. 35, no. 3, pp. 314–341, 2021.

[26] L. Wang, Y. Hu, and C. Liu, "Online adaptation of neural network models by modified extended kalman filter for customizable and transferable driving behavior prediction," 2021, *arXiv:2112.06129*.

[27] W. Zhan et al., "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," 2019, *arXiv:1910.03088*.

[28] G. Markkula et al., "Defining interactions: A conceptual framework for understanding interactive behaviour in human and automated road traffic," *Theor. Issues Ergonom. Sci.*, vol. 21, no. 6, pp. 728–752, 2020.