Cooperative Driving at Unsignalized Intersections Using Tree Search

Huile Xu, Yi Zhang, *Member, IEEE*, Li Li, *Fellow, IEEE*, and Weixia Li

***Abstract*— In this paper, we propose a new cooperative driving strategy for connected and automated vehicles (CAVs) at unsignalized intersections. Based on the tree representation of the solution space for the passing order, we combine Monte Carlo tree search (MCTS) and some heuristic rules to find a nearly global-optimal passing order (leaf node) within a very short planning time. Testing results show that this new strategy can keep a good tradeoff between performance and computation flexibility.**

***Index Terms*— Connected and automated vehicles (CAVs), cooperative driving, unsignalized intersection, Monte Carlo tree search (MCTS).**

I. Introduction

CONNECTED and Automated Vehicles (CAVs) are believed to be a key role in the next-generation transportation systems [1], [2]. With the aid of vehicle-to-vehicle (V2V) communication, CAVs can share their driving states (position, velocity, etc.) and intentions with adjacent vehicles [3], [4] to better coordinate their motions to alleviate traffic congestion and improve traffic safety [5], [6].

In the last decade, various strategies had been proposed to make optimal coordination for CAVs at a typical driving scenario: unsignalized intersection. It is pointed out in [7] and [8] that the key problem is to determine the optimal order of CAVs that passed the intersection. As summarized in [9], there are two kinds of cooperative driving strategies, planning based and ad hoc negotiation based, for determining the passing order.

Planning based strategies aim to enumerate all possible passing orders to find the globally optimal solution [10]. There are two equivalent formulations of the problem. Most state-of-the-art studies formulate the problem as a mixed integer linear programming (MILP) problem of vehicles' passing time scheduling [1], [11]. The objective is usually set to minimize the total delay of all CAVs. Li and Wang showed that we can also view this problem as a tree search problem [8]. Each tree node indicates a special (partial) passing order. The equivalent objective is to find the leaf node corresponds to the minimum total delay of all CAVs [7], [8]. It was shown in [12] that some planning based strategies work well for ramp metering scenarios. However, the time to enumerate all the nodes increases sharply as the number of vehicles increases, especially for unsignalized intersection scenarios. This problem hinders their applications in practice.

Ad hoc negotiation based strategies aim to find an acceptable passing order using some heuristic rules within a very short time. For example, Stone *et al.* proposed autonomous intersection management (AIM) cooperative driving strategy which divides the intersection into grids (resources) and assigns these grids to CAVs in a roughly First-In-First-Out (FIFO) manner [13], [14]. This strategy has several variations, including reservation strategy [15]. Based on the concept of virtual vehicles [16], Xu *et al.* proposed a spanning tree algorithm for a projected virtual platoon to determine a conflict-free passing order [17]. Other examples such as conflict graph can be found in [12] and [18]. However, as shown in [9], the passing orders found by ad hoc negotiation based strategies were not good enough in many situations.

To keep a good tradeoff between performance and computation flexibility, we propose a new cooperative driving strategy based on the tree representation of the solution space for the passing order. Its key idea is to use the limited planning time to explore the nodes that are potential to be the optimal solution. To this end, we combine Monte Carlo tree search (MCTS) and some heuristic rules to accelerate the searching process, since the solution space of this problem has special structures to be exploited. Testing results show that we can find a nearly global-optimal passing order within a short enough planning time.

It should be pointed out that [8] pioneered the idea that the optimization problem of finding the optimal passing order of vehicles can be transferred into a tree search problem. However, the work solved the tree search problem through the enumeration method since the considered number of vehicles is small, which makes it difficult to be applied to scenarios with a large number of vehicles. Inspired by the work, this paper aims to further explain how to solve the tree search problem effectively. Its main contribution is that it demonstrates that a nearly global-optimal passing order can be found within a very short planning time by combining the MCTS and some heuristic rules.

To give a better presentation of our finding, the rest of this paper is arranged as follows. *Section II* formulates the problem

Manuscript received February 9, 2019; revised April 22, 2019 and June 25, 2019; accepted September 6, 2019. Date of publication September 25, 2019; date of current version October 30, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1600600. The Associate Editor for this article was X. Ban. *(Corresponding author: Li Li.)*

H. Xu, L. Li, and W. Li are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: hl-xu16@mails.tsinghua.edu.cn; li-li@tsinghua.edu.cn; wx-li11@mails.tsinghua.edu.cn).

Y. Zhang is with the Department of Automation, BNRist, Tsinghua University, Beijing 100084, China, and also with the Tsinghua-Berkeley Shenzhen Institute (TBSI), Shenzhen 518055, China (e-mail: zhyi@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TITS.2019.2940641

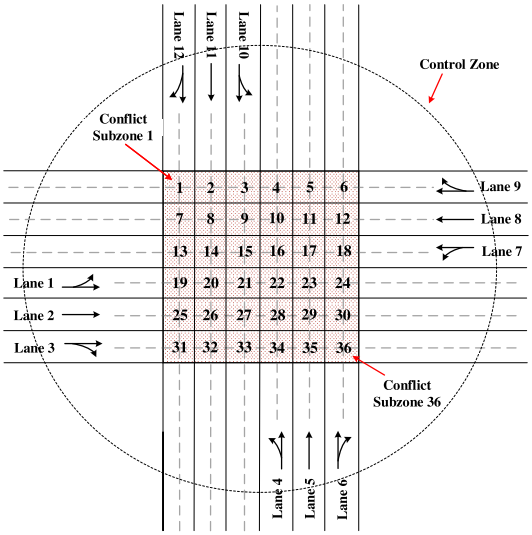

Fig. 1. A typical intersection scenario.

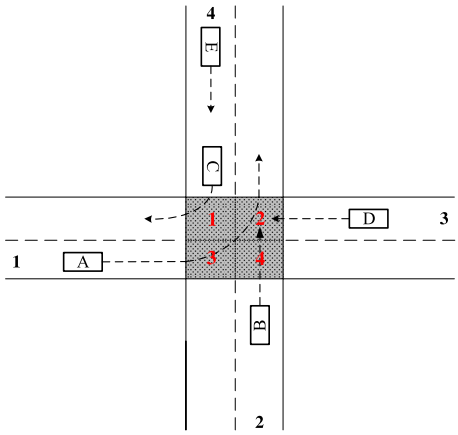

Fig. 2. An intersection scenario with 5 vehicles.

and briefly reviews the existing strategies. *Section III* presents the new strategy. *Section IV* validates the effectiveness of the proposed strategy via numerical testing results. Finally, *Section V* gives concluding remarks.

II. Problem Formulation

Fig. 1 shows a typical intersection scenario with multiple lanes in each leg. The area within the circle is called the control zone, and the shadow area is called the conflict zone where lateral collisions might happen. According to the geometry of the intersection, the conflict zone can be further divided into several conflict subzones. For example, the conflict zone in Fig. 1 is divided into 36 conflict subzones which are named from Conflict Subzone 1 to Conflict Subzone 36.

We assign each vehicle that enters the control zone a unique identity V_i. We also use the set Z_i to denote the conflict subzones that V_i will pass through. For example, $Z_i = \{4, 1\}$ means V_i will pass through Conflict Subzone 4 and Conflict Subzone 1 in sequence.

To simplify the problem, we adopt the following assumptions:

- Each vehicle instantly and thoroughly shares its driving states (position, velocity, etc.) and intentions with other vehicles via vehicle-to-vehicle (V2V) communication.
- Changing lane maneuver is prohibited in the control zone to ensure vehicle safety.
- Similar to [15] and [19], the velocities of vehicles are constant when passing through the conflict zone.

The cooperative driving strategy aims to minimize the total delay of vehicles by scheduling the velocity and acceleration profiles of all vehicles [20]. So, we can get the following optimization problem

$$\min \ J = \sum_{i=1}^{n} (t_{assign,i,Z_i(1)} - t_{min,i,Z_i(1)}), \tag{1}$$

where $t_{assign,i,z}$ is the desired arrival time to the conflict subzone z for V_i, $t_{min,i,z}$ is the minimum arrival time to the conflict subzone z when V_i travels at the maximum velocity and the maximum acceleration, $Z_i(1)$ is the first element in the set Z_i, n is the number of vehicles in the control zone.

To directly attack Problem (1) often leads to a MILP problem whose computation time increases exponentially with the increase of the number of vehicles [11], [12].

Noticing that the traffic efficiency mainly depends on the passing order of vehicles [9], we can formulate the whole problem as a tree search problem in the solution space that consists of all possible passing orders. Each leaf node represents a passing order of vehicles which can also be denoted as a string [8]. For example, string CAB means vehicle C, vehicle A, and vehicle B enter the conflict zone sequentially.

It should be pointed out that the passing order indicates the priorities of vehicles. If two vehicles pass through the same conflict zone, the vehicle which ranks higher in the passing order will have the priority of passing through the conflict zone first. Take the intersection scenario shown in Fig. 2 as an example. The partial passing order BD means that V_D yields V_B when passing through the Conflict Subzone 2 and adjusts its arrival time according to the arrival time of V_B. If the partial passing order is BC, the arrival time of V_C to the Conflict Subzone 1 will be its minimum arrival time since V_B and V_C pass through different conflict subzones. In other words, the objective function values of partial passing orders BC and CB are the same. Each vehicle greedily determines its fastest arrival times to each conflict subzone according to the passing order. Thus, two vehicles without confliction can pass the intersection at the same time, and a vehicle might enter into the conflict zone earlier than some vehicles that are in front of it in the passing order.

Let us take the intersection scenario shown in Fig. 2 as an example to explain how to build the tree representation of the solution space gradually. At first, we set the passing order in the root node to be empty. Then, each direct child node of the root node (in the second layer) refers to one index symbol that indicates the first vehicle in a special passing order. The nodes in the third layer refer to one string consisting of two indices symbols that indicate the first two vehicles in a special passing order. Similarly, the child nodes expand their

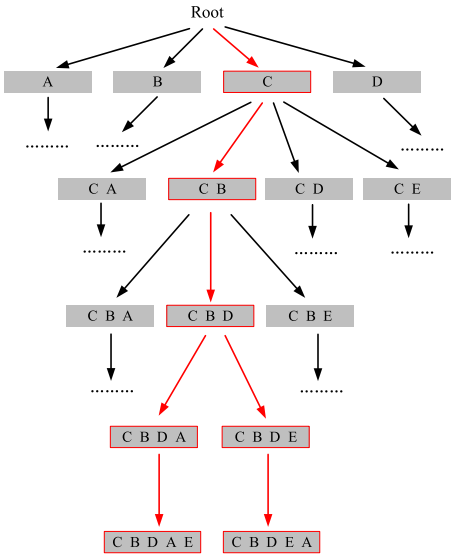

Fig. 3. The solution tree stemmed from the intersection scenario shown in Fig. 2. The leaf nodes in the bottom layer represent the complete passing orders for all vehicles.

child nodes, and all possible passing order are generated as leaf nodes in the bottom layer of the solution tree as shown in Fig. 3.

If a (partial) passing order is given, the desired arrival times for all the vehicles that has been covered in this (partial) passing order can be directly derived by the following Passing Order to Trajectory Interpretation Algorithm. Our objective turns to seek the leaf node that corresponds to the shortest total delay. Moreover, the total delay values of leaf nodes can be used to evaluate the potential of their parent nodes in a backpropagation way. This method provides us a chance to find a nearly global-optimal leaf node but only search a small part of the whole tree.

Algorithm 1 Passing Order to Trajectory Interpretation

Require: A (partial) passing order P

Ensure: The total delay J of the covered vehicles and their arrival times t_{assign}

1: **for** each $i \in [1, length(P)]$ **do**
2: **for** each $z \in Z_i$ **do**
3: V_j is the last vehicle that passed through subzone z
4: $t_{assign,P(i),z} = max(t_{min,P(i),z}, t_{max,z} + \Delta_{j,a})$
5: **end for**
6: Adjust $t_{assign,P(i),z}$ according to the constraint: the velocity of V_i in the conflict zone is constant.
7: **for** each $z \in Z_i$ **do**
8: $t_{max,z} = t_{assign,P(i),z}$
9: **end for**
10: **end for**
11: $J = \sum_{i=1}^{length(P)} t_{assign,P(i),Z_i(1)}$

In **Algorithm 1**, $P(i)$ is the ith element in the input (partial) passing order, $t_{max,z}$ is the largest arrival time that the subzone z has been occupied. $\Delta_{j,a}$ is the minimum safety gap between two consecutive vehicles passing through the same subzone. Obviously, the time complexity of **Algorithm 1** is $O(n)$. A detailed explanation of **Algorithm 1** can be found in our previous report [10].

III. MCTS Based Cooperative Driving Strategies

It is usually impossible to expand all the nodes of the solution tree within the limited computation budget, when there are lots of vehicles in the control zone. In this paper, we use MCTS + heuristic rules to select nodes with the potential to be the optimal solution. The recent success of the MCTS method in the game of Go shows it is an effective way to deal with such problems [21], [22].

A. The Classical MCTS Based Strategy

In MCTS, each node in the formulated tree will be assigned a score to evaluate its potential. The score of a leaf node is equal to the total delay of its corresponding passing order. MCTS uses these scores to determine which branch of tree to explore.

Generally, MCTS gradually builds a search tree in an iteration way. One iteration consists of four steps: selection, expansion, simulation, and backpropagation [23]; see Fig. 4.

1) Selection: Starting at the root node, we select the most urgent expandable node based on the following policy [24]

$$\arg\max_i \; Q_i + C\sqrt{\frac{\ln n}{n_i}}, \tag{2}$$

where Q_i is the score of child node i and the value of Q_i is within $[0, 1]$. n is the number of times the current node has been visited, n_i is the number of times child node i has been visited, and C is a weighting parameter. The child node with the largest total score is selected. Here, an expandable node refers to a node that is not a leaf node and has unvisited child nodes.
This child node selection policy is suggested in the field of computer Go and is called UCB1 [24]. The first term in the equation encourages to select the child node that is currently believed to be optimal, while the second term encourages to explore more child nodes.
2) Expansion: We randomly select one unvisited child node of the most urgent expandable node to be a new node that is added to the tree.
3) Simulation: We run several rollout simulations to determine a complete passing order based on the partial passing order represented by the current new node to evaluate the potential of the new node.
The classical MCTS randomly samples and adds the uncovered vehicles into the passing order string one by one, until we find a complete passing order string and reach the maximum depth of the tree from the current new node without branching [22]. For example, when we apply random sampling policy to the node CB shown in Fig. 4, we can randomly expand a direct child node in its next layer; say node CBA. The node CBA will be

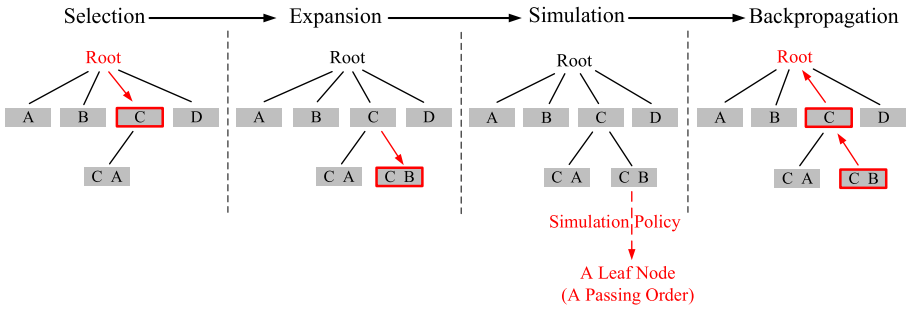

Fig. 4. One iteration of the MCTS based cooperative driving strategy.

further expanded by repeating such a process until a leaf node (e.g., node CBADE) is reached. Finally, the partial passing order will be evaluated by its simulated off-spring leaf node (passing order). Sometimes, the generated passing order is invalid, because it may violate the prohibition of lane change, such solutions will be discarded after check.

After simulation, we update the scores of the current new node as follows:

i Apply **Algorithm 1** to calculate the total delay $\bar{J}_i$ of the partial passing order corresponds with the current new node.

ii Apply **Algorithm 1** to calculate the total delay $\hat{J}_i$ of the passing order corresponds with the best off-spring node of the current new node via simulation.

iii Since the score of the current new node Q_i belongs to the interval [0, 1], we normalize $\bar{J}_i$ and $\hat{J}_i$ into [0, 1] using

$$q_i = 1 - (J_i - J_{i,\min})/(J_{i,\max} - J_{i,\min}), \quad (3)$$

where $J_{i,\max}$ is the maximum total delay among the sibling nodes of the node i, and $J_{i,\min}$ is the minimum total delay among the sibling nodes of the node i.

iv Calculate the score Q_i of the current new node as

$$Q_i = \omega \bar{q}_i + (1 - \omega)\hat{q}_i, \quad (4)$$

where ω is a weighting parameter.

4) Backpropagation: The simulation result is backpropagated through the selected nodes to update the scores of all its parent nodes.

During the building process of the search tree, the state-of-the-art best passing order is continuously updated. As soon as the computation budget is reached, the search terminates and returns the state-of-the-art best passing order. The planned arrival times of vehicles can be determined by using **Algorithm 1**. The velocity and acceleration profiles of each vehicle plan will be finally calculated by using the motion planning method proposed in [20].

We can see that the performance of the proposed strategy is influenced by the choice of the parameters including the maximum search time and two weighting parameters C and ω. We will discuss how to choose these parameters in *Section IV* below.

B. The MCTS + Heuristic Rules

As aforementioned, the classical MCTS strategy uses random sampling to generate a leaf node (a passing order) in the simulation step. However, because of the huge number of possible passing orders, the passing orders generated by random sampling cannot help us quickly capture the real potential of a node during simulation.

Thus, we propose the following heuristic rules to help decide which nodes (vehicles) should be expanded (added into the candidate passing order string) during simulation. Heuristic rule 1 helps to quickly prune the invalid passing order [8]. Heuristic rule 2 determines the vehicle among the candidates to be chosen.

1) For the vehicles on the same lane, the vehicle which is the closest to the conflict zone should be added earlier than other vehicles since changing lane maneuver is prohibited.
2) For the vehicles passing through the same conflict subzone, the vehicle with a less desired arrival time should be added earlier.

The simulation step can be summarized as **Algorithm 2**. We can see that the classic MCTS applies random sampling in both expansion and simulation steps; while our MCTS + heuristic rules applies random sampling only in expansion step.

Algorithm 2 Heuristic Simulation Policy

Require: Locations and velocities of all vehicles

Ensure: A possible passing order

1: Among all uncovered vehicles, we select the vehicles which are the closest to the conflict zone in each lane as candidate vehicles and calculate their arrival times to all conflict subzones.

2: If there exists a candidate vehicle whose arrival times to all conflict subzones are all the smallest, we add it into the passing order string. If not, we randomly select one vehicle among all candidate vehicles and add it into the passing order string.

3: Then we repeat the steps 1 and 2 until a complete passing order string is generated.

4: The objective value (1) of the generated passing order can be easily derived by **Algorithm 1** and denoted as $\hat{J}_i$.

The ad hoc negotiation based strategies organize all the vehicles according to the FIFO principle. In contrast, the new simulation policy tends to organize just a part of vehicles (the vehicles uncovered in the current partial passing order) according to the FIFO strategy. This trick helps to avoid the convergence to a over-greedy solution.

C. Trajectory Planning

After determining the desired arrival times to all conflict subzones, we need to address the problem of optimally controlling the accelerations of vehicles so as to enable them to reach the conflict subzones at the desired time and velocity. The dynamics of each vehicle can be represented by a state function

$$\dot{x_i} = f(t, x_i, u_i), \quad x_i(t) = [p_i(t), v_i(t)], \tag{5}$$

where $x_i(t)$ and $u_i(t)$ are the state and control input of V_i at time t, $p_i(t)$ and $v_i(t)$ are the position and velocity of V_i at time t. It has been pointed out in [25] that for the trajectory planning problems in the traffic flow field, we usually use a simplified dynamics model since it is hard to solve the nonlinear optimization problem and the modeling errors would not contribute too much for the loss of objective values. Thus, similar to relevant works [26] and [27], the second order dynamics model is used and we have

$$\dot{p}_i(t) = v_i(t), \tag{6}$$

$$\ddot{p}_i(t) = u_i(t). \tag{7}$$

Then, the acceleration profile of each vehicle can be calculated through the following optimization problem and an analytical solution for the problem can be found in [20]

$$\min_{u_i(t)} \ \frac{1}{2}\int_{t_i^0}^{t_{assign,i,Z_i(1)}} u_i^2(t) \tag{8a}$$

$$\text{subject to } u_{\min} \le u_i(t) \le u_{\max} \tag{8b}$$

$$v_{\min} \le v_i(t) \le v_{\max} \tag{8c}$$

$$\dot{p}_i(t) = v_i(t),\ \ddot{p}_i(t) = u_i(t) \tag{8d}$$

$$p_i(t_i^0) = 0,\ v_i(t_i^0) = v_i^0 \tag{8e}$$

$$p_i(t_{assign,i,Z_i(1)}) = p_i^0 \tag{8f}$$

$$v_i(t_{assign,i,Z_i(1)}) = v_c \tag{8g}$$

$$t \in [t_i^0, t_{assign,i,Z_i(1)}] \tag{8h}$$

where t_i^0 is the initial time point, p_i^0 is the initial position, v_i^0 is the initial velocity, and v_c is the constant velocity in the conflict zone.

It should be pointed out that different vehicle dynamics and different objectives may lead to different acceleration profiles and may influence the performance of the proposed strategy. However, it is not within the scope of this paper and interested readers can refer to [25] and [28] for further details on the trajectory planning and speed harmonization.

IV. Simulation Results

A. Simulation Settings

We design three experiments to determine the best parameter set for the new cooperative driving strategy and compare it with some classical ones. These experiments are conducted for the intersection with three lanes in each leg shown in Fig. 1. The mandatory signs stipulate the permitted directions for each lane. In the simulation, for the leftmost lane of each direction, half of the vehicles will turn left while half of vehicles will go straight; similarly, for the rightmost lane, half of the vehicles will turn right while half of vehicles will go straight; vehicles on the middle lane are only allowed to go straight.

According to the geometry of the intersection, the conflict zone is further divided into 36 subzones. The vehicles arrival is assumed to be a Poisson process. We vary the mean value of this Poisson process to test the performance of the proposed strategy under different traffic demands. The vehicles arrival rates at all lanes are the same unless otherwise specified. It should be pointed out that we had tested other intersections with different road geometries and various vehicle arrival patterns, but the conclusions remain unchanged.

To accurately describe the total delays of vehicles, we adopt the point-queue model in the simulation [9], [29]. The model assumes vehicles travel in free flow state until they get to the boundary of the intersection we study. If the preceding vehicle leaves enough spaces, the first vehicle in the point-queue will dequeue and enter the intersection. Otherwise, it will stay in the virtual queue. Each lane has an independent point-queue.

In this paper, we reschedule the passing order of all the vehicles within the control zone every 2 seconds. As suggested in [9], we set the minimum safety gap between two consecutive vehicles passing through the same subzone as a slightly enlarged constant as

$$\Delta_{j,a} = \begin{cases} 1.5s & a = 1 \text{ (go straight)} \\ 2s & a = 2 \text{ (turn left)} \\ 1.5s & a = 3 \text{ (turn right)} \end{cases} \tag{9}$$

to avoid the collisions caused by position measurement errors and communication delay.

All experiments are implemented using C++ language on a Visual Studio platform in a personal computer with an Intel i7 CPU and a 16GB RAM.

B. The Choice of Parameters

In this paper, we consider two performance indices: the delay J of the given n vehicles and the traffic throughput (the number of vehicles that has passed the intersection control zone) within a given time interval to compare different cooperative driving strategies. Specially, we highlight the decreased ratio of the total delay if being compared with the baseline solution that is gotten by the FIFO strategy

$$\eta = \frac{J_{FIFO} - J_{MCTS}}{J_{FIFO}}, \tag{10}$$

where J_{FIFO} is the objective value of the FIFO passing order, and J_{MCTS} is the objective value of the best passing order from the MCTS based strategy.

To determine the best parameter setting of the new MCTS + heuristic rules, we first fix the time budget as 0.1 s and vary ω and C from 0 to 1. To better understand the performance of the strategy under different traffic conditions, we vary the vehicle

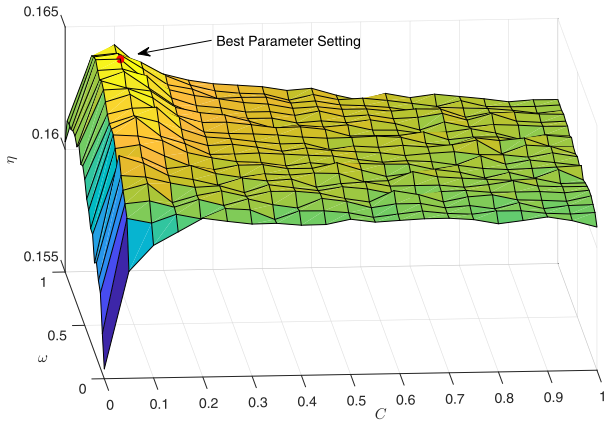

Fig. 5. The improvement rate of the MCTS based strategy with different parameter settings for the intersection shown in Fig.1.

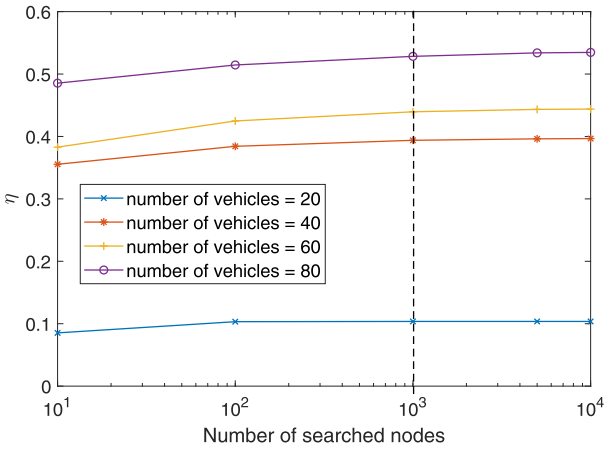

Fig. 6. The results of the improvement rates with respect to the number of searched nodes for the intersection shown in Fig.1.

arrival rate to generate a series of intersection scenarios with different number of vehicles.

Fig. 5 gives the improvement rates for the intersection shown in Fig.1 with 30 vehicles. We can see that a significant improvement can be achieved even with the worst parameter setting. The parameter C and ω are not so critical but may still influence the balance between exploitation and exploration, partly because we use heuristic rules in simulation step to reduce the influence of random sampling. We further study the scenarios with other numbers of vehicles and the results are all similar. Thus, in the rest of this paper, we set $\omega = 0.85$ and $C = 0.05$.

Then, to determine an appropriate time budget, we vary the time limits of tree search. To eliminate the influence of the computing power of the device, we examine the improvement rates with respect to the number of nodes that has been searched.

It can be seen from Fig. 6 that the improvement rate increases significantly when the number of searched nodes increases from 10 to 1000. However, the improvement rate soon becomes saturated after that. Thus, we believe that the proposed strategy can obtain a good enough passing order

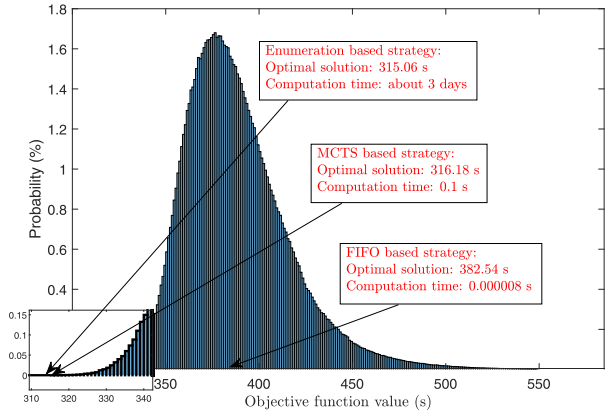

Fig. 7. The histogram of all solution values for a single lane intersection scenario with 20 vehicles.

through searching 1000 nodes. For most intersection scenarios, 1000 nodes can be searched within 0.1 s in our personal computer, so we set the maximum search time as 0.1 s for the following experiments.

When the number of vehicles increases, the coordination performance of the FIFO based strategy declines. At the same time, the proposed MCTS based strategy always can find a nearly global-optimal solution no matter how many vehicles there are. So, although the objective value (total delay) increases with the number of vehicles, the improvement rate of the objective value also increases with the number of vehicles because of inefficient coordination of the FIFO based strategy.

C. Comparisons of Different Cooperative Driving Strategies

To further clarify the difference between the FIFO strategy and our new strategy, we study a typical intersection scenario with single lane in each leg and 20 vehicles. We calculate the objective values for all the valid solutions (passing orders) and plot them in a histogram manner; see Fig. 1.

It is clear that the solution found by the MCTS based strategy is nearly the same as the global optimal solution found by the enumeration based strategy, while the computation time of the MCTS based strategy is much less. For the FIFO based strategy, the computation time is the least, but the solution is far away from the optimal solution. The solution found by the MCTS based strategy ranks 648th in the nearly 10 billion solutions; while the solution of the FIFO based strategy ranks 4563421793rd.

We then carry out another comparison for the intersection shown in Fig. 1, where the average arrival rate is varied to explore the influence of different traffic demands. For each arrival rate, we simulate a 20-minute traffic process. It is obvious that our new strategy further reduces the average delay and improves the traffic throughput in all situations.

D. The Influence of Asymmetric Traffic Demands

The traffic demands are mainly reflected in two aspects including the arrival rate of each lane and origin-destination

TABLE I
COMPARISON RESULTS OF DIFFERENT COOPERATIVE DRIVING STRATEGIES

Arrival rate veh/(lane*h)	Strategies	Average delay (s)	Traffic Throughput (veh)
100	FIFO	0.5864	425
	MCTS	0.2928	425
200	FIFO	4.0776	804
	MCTS	0.5898	804
300	FIFO	39.8313	1095
	MCTS	1.1407	1168

* The computation time of the MCTS based strategy is 0.1s.

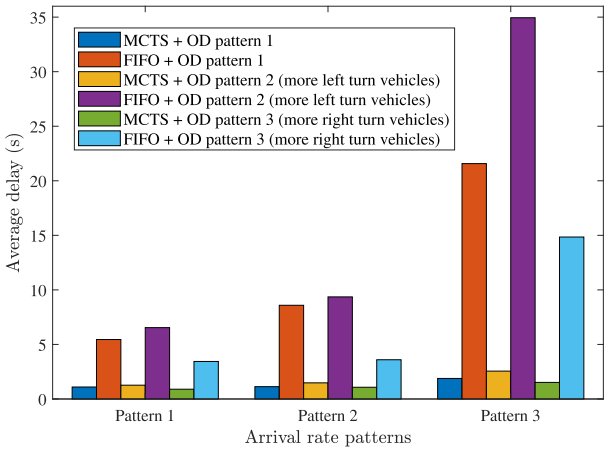

Fig. 8. The comparison results of the MCTS based strategy and FIFO based strategy under asymmetric traffic demands.

(OD) pattern. To analyze the influence of asymmetric traffic demands on the proposed method, we set the arrival rates of the lanes in some directions as larger values than other directions and vary the turn ratios of some lanes to generate asymmetric demands.

In this paper, the arrival rate pattern i means that the traffic flow rates of i directions are 300 veh/h/lane, while the traffic flow rates of the remaining directions are 150 veh/h/lane. In addition, we design three kinds of OD patterns. The OD pattern 1 is the pattern we introduced in the simulation settings; in OD pattern 2, for the leftmost lane of each direction, 80% of the vehicles will turn left while 20% of vehicles will go straight, and other settings are the same as the OD pattern 1; in OD pattern 3, for the rightmost lane of each direction, 80% of the vehicles will turn right while 20% of vehicles will go straight, and other settings are the same as the OD pattern 1. The comparison results of the MCTS based strategy and FIFO based strategy under asymmetric traffic demands are shown in Fig. 8.

It is clear that compared with the FIFO based strategy, the proposed strategy can significantly reduce the average delay no matter which arrival rate pattern it is. Then, it can be observed that vehicles turning left cause greater delay since they occupy more merging subzones. The impacts of different arrival rate patterns and OD patterns on the FIFO based strategy are obvious. In contrast, the MCTS based strategy still can perform well under all situations.

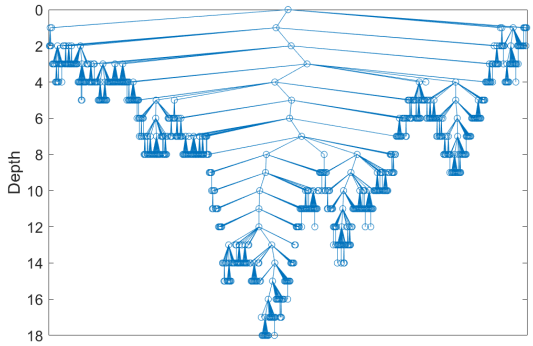

Fig. 9. The structure of a search tree for an intersection scenario with 50 vehicles.

E. A Further Look Into the Structure of the Obtained Search Tree

Fig. 9 shows the formulated search tree of our new strategy for an intersection scenario with 50 vehicles. Similar to the classical MCTS strategy, our new strategy tends to first find some promising branches (partial passing orders) of the tree and spends most search time to further explore these branches. However, the search tree generated by the classical MCTS strategy contains much more unnecessary leaf nodes. In contrast, when the heuristic rules are introduced, only a very small number of leaf nodes will be finally reached. For this case, although there are more than 10^{46} possible passing orders, only about two thousands passing orders are explored by our new strategy within 0.1 s. This difference explains why the classical MCTS needs much more time to find a good enough passing order.

Different from some random algorithms, the heuristic rules help us to generate some passing orders based on human knowledge, then MCTS will try to adjust the currently optimal passing order to find a better passing order. Thus, the proposed strategy can always generate a reasonable and better passing order than that based entirely on human knowledge.

V. FURTHER DISCUSSION

This section briefly analyzes the capacity of the intersection to show the feasibility of the proposed strategy when dealing with actual vehicles and real-world roadway traffic. Since vehicles arrival is assumed to be a Poisson process and the arrivals of different OD patterns are independent. It is easy to derive the probability of the event that the nth vehicle entering the control zone is from the OD pattern i is

$$P_i^n = \frac{\lambda_i}{\lambda_{total}}, \tag{11}$$

where λ_i is the arrival rate of OD pattern i and λ_{total} is the sum of arrival rates of all OD patterns.

For the convenience of the analysis, some notations are made. Depending on its OD pattern, every vehicle belongs to only one of the following three subsets: 1) $\mathcal{R}$ contains all

TABLE II
THE PERFORMANCE OF DIFFERENT COOPERATIVE DRIVING STRATEGIES WHEN THE TRAFFIC DEMAND IS THE INTERSECTION CAPACITY

Strategy	Throughput (veh/h)	Delay (s)
MCTS	3897	4.28495
FIFO	3333	37.58254

right-turn OD patterns, 2) $\mathcal{L}$ contains all left-turn OD patterns, 3) $\mathcal{S}$ contains all straight OD patterns. $\mathcal{C}_i$ contains all OD patterns that have conflict with OD pattern i.

To simplify the analysis, we assume that the value of the minimum safety gap Δ_t only depends on the relationship between two consecutive vehicles in the passing order. For example, consider a passing order ab, if there is no conflict between the OD patterns of V_a and V_b, then $\Delta_t = 0$; if their OD patterns conflict with each other, then Δ_t is t_r (V_a turn right), t_l (V_a turn left) and t_s (V_a go straight). Thus, the probability distribution of the value of Δ_t is

$$P(\Delta_t = t_r) = \sum_{i\in\mathcal{R}}(P_i^{n-1}\sum_{k\in\mathcal{C}_i}(P_k^n)), \tag{12}$$

$$P(\Delta_t = t_l) = \sum_{i\in\mathcal{L}}(P_i^{n-1}\sum_{k\in\mathcal{C}_i}(P_k^n)), \tag{13}$$

$$P(\Delta_t = t_s) = \sum_{i\in\mathcal{S}}(P_i^{n-1}\sum_{k\in\mathcal{C}_i}(P_k^n)), \tag{14}$$

$$P(\Delta_t = 0) = 1-P(\Delta_t = t_r)-P(\Delta_t = t_l)-P(\Delta_t = t_s). \tag{15}$$

Then according to the queueing theory and the analysis in [30], the expected service time $E(S)$ is

$$E(S) = t_r \times P(\Delta_t = t_r)+t_l\times P(\Delta_t = t_l)+t_s \times P(\Delta_t = t_s), \tag{16}$$

and the intersection capacity is

$$C = \frac{3600}{E(S)}. \tag{17}$$

Take the intersection shown in Fig. 1 as an example, we set the arrival rates of all lanes to the same value and set $t_r = 2.5s$, $t_l = 3s$, and $t_s = 2.5s$. Then, according to the simulation settings, the intersection capacity is 4098 veh/h. To assess the feasibility of the proposed strategy, we set the total arrival rates to the value of capacity, and Table II shows the comparison results.

It is clear that even the traffic demand is close to the maximum saturation flow rate, the proposed strategy still performs well, and the throughput is very close to the intersection capacity, which shows that the proposed strategy is promising to be applied in practice.

Concerning the vehicular dynamics, most of the existing works in this field did not take the vehicular dynamics into account. To compare the results with relevant works, most of the assumptions used in this paper are set as the same as them. In addition, we have tested the effect of cellular automata modeling approach in our previous work [9]. We found that the difference between two representative trajectory planning algorithms (the virtual vehicle mapping algorithm and cellular automata algorithm) is small and could be omitted and the traffic efficiency mainly depends on the passing order.

VI. CONCLUSION

In this paper, we propose a cooperative driving strategy that combines Monte Carlo simulation and heuristic rule simulation to accelerate the search of the passing order. This new method can quickly learn the tree structure knowledge of the given scenario and find a nearly optimal solution with a short time. Although we only discuss the schedule of vehicles at unsignalized intersections, this method can be easily adapted to other scenarios (e.g., ramping areas and working zones). We are currently building several automated vehicle prototypes so that we can test our new strategy in field studies in the near future.

REFERENCES

[1] P. Li and X. Zhou, "Recasting and optimizing intersection automation as a connected-and-automated-vehicle (CAV) scheduling problem: A sequential branch-and-bound search approach in phase-time-traffic hypernetwork," *Transp. Res. B, Methodol.*, vol. 105, pp. 479–506, Nov. 2017.

[2] Q. Guo, L. Li, and X. J. Ban, "Urban traffic signal control with connected and automated vehicles: A survey," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 313–334, Apr. 2019.

[3] L. Li, D. Wen, and D. Y. Yao, "A survey of traffic control with vehicular communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 425–432, Feb. 2014.

[4] T. Sukuvaara and P. Nurmi, "Wireless traffic service platform for combined vehicle-to-vehicle and vehicle-to-infrastructure communications," *IEEE Wireless Commun.*, vol. 16, no. 6, pp. 54–61, Dec. 2009.

[5] S. Feng, Y. Zhang, S. E. Li, Z. Cao, H. X. Liu, and L. Li, "String stability for vehicular platoon control: Definitions and analysis methods," *Annu. Rev. Control*, vol. 47, pp. 81–97, Mar. 2019.

[6] J. Ding, L. Li, H. Peng, and Y. Zhang, "A rule-based cooperative merging strategy for connected and automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8767981

[7] S. I. Guler, M. Menendez, and L. Meier, "Using connected vehicle technology to improve the efficiency of intersections," *Transp. Res. C, Emerg. Technol.*, vol. 46, pp. 121–131, Sep. 2014.

[8] L. Li and F. Y. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Trans. Veh. Technol.*, vol. 55, no. 6, pp. 1712–1724, Nov. 2006.

[9] Y. Meng, L. Li, F.-Y. Wang, K. Li, and Z. Li, "Analysis of cooperative driving strategies for nonsignalized intersections," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 2900–2911, Apr. 2018.

[10] H. Xu, S. Feng, Y. Zhang, and L. Li, "A grouping-based cooperative driving strategy for CAVs merging problems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6125–6136, Jun. 2019.

[11] E. R. Müller, R. C. Carlson, and W. K. Junior, "Intersection control for automated vehicles with MILP," *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 37–42, 2016.

[12] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[13] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst.*, vol. 2, 2004, pp. 530–537.

[14] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[15] M. Choi, A. Rubenecia, and H. H. Choi, "Reservation-based cooperative traffic management at an intersection of multi-lane roads," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 456–460.

[16] A. Uno, T. Sakaguchi, and S. Tsugawa, "A merging control algorithm based on inter-vehicle communication," in *Proc. IEEE/IEEJ/JSAI Int. Conf. Intell. Transp. Syst.*, Oct. 1999, pp. 783–787.

[17] B. Xu *et al.*, "Distributed conflict-free cooperation for multiple connected vehicles at unsignalized intersections," *Transp. Res. C, Emerg. Technol.*, vol. 93, pp. 322–334, Aug. 2018.
[18] C. Liu, C. Lin, S. Shiraishi, and M. Tomizuka, "Distributed conflict resolution for connected autonomous vehicles," *IEEE Trans. Intell. Veh.*, vol. 3, no. 1, pp. 18–29, Mar. 2018.
[19] Y. Zhang, A. A. Malikopoulos, and C. G. Cassandras, "Decentralized optimal control for connected automated vehicles at intersections including left and right turns," in *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, Dec. 2017, pp. 4428–4433.
[20] A. A. Malikopoulos, C. G. Cassandras, and Y. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, Jul. 2018.
[21] M. Enzenberger, M. Müller, B. Arneson, and R. Segal, "Fuego—An open-source framework for board games and go engine based on Monte Carlo tree search," *IEEE Trans. Comput. Intell. AI in Games*, vol. 2, no. 4, pp. 259–270, Dec. 2010.
[22] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[23] 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.
[24] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 2006, pp. 282–293.
[25] L. Li and X. Li, "Parsimonious trajectory design of connected automated traffic," *Transp. Res. B, Methodol.*, vol. 119, pp. 1–21, Jan. 2019.
[26] Y. Zhang and C. G. Cassandras, "A decentralized optimal control framework for connected automated vehicles at urban intersections with dynamic resequencing," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 217–222.
[27] A. A. Malikopoulos, S. Hong, B. B. Park, J. Lee, and S. Ryu, "Optimal control for speed harmonization of automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2405–2417, Jul. 2019.
[28] J. Ma *et al.*, "Freeway speed harmonization," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 78–89, Mar. 2016.
[29] X. Ban, J.-S. Pang, H. X. Liu, and R. Ma, "Continuous-time point-queue models in dynamic network loading," *Transp. Res. B, Methodol.*, vol. 46, no. 3, pp. 360–380, 2012.
[30] C. Yu, W. Sun, H. X. Liu, and X. Yang, "Managing connected and automated vehicles at isolated intersections: From reservation-to optimization-based methods," *Transp. Res. B, Methodol.*, vol. 122, pp. 416–435, Apr. 2019.

Huile Xu received the B.S. degree from Chongqing University, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University, China. His research interests include cooperative driving and intelligent vehicles.

Yi Zhang (S'01–M'04) received the B.S. and M.S. degrees from Tsinghua University, China, in 1986 and 1988, respectively, and the Ph.D. degree from the University of Strathclyde, U.K., in 1995. He is currently a Professor in control science and engineering with Tsinghua University. His current research interests include intelligent transportation systems, intelligent vehicle-infrastructure cooperative systems, analysis of urban transportation systems, urban road network management, traffic data fusion and dissemination, urban traffic control and management, advanced control theory and applications, advanced detection and measurement, systems engineering, and so on.

Li Li (S'05–M'06–SM'10–F'17) is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China, where he is involved in the fields of complex and networked systems, intelligent control and sensing, intelligent transportation systems, and intelligent vehicles. He has published over 90 SCI indexed international journal articles and over 70 international conference articles as a first/corresponding author. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, a member of the Editorial Advisory Board for *Transportation Research Part C: Emerging Technologies*, and a member of the Editorial Board for *Transport Reviews*.

Weixia Li received the B.S. degree in automation from Beijing Jiaotong University, Beijing, China, in 2007, and the Ph.D. degree in control science and technology from Tsinghua University, Beijing, in 2017. From 2015 to 2016, she was a Visiting Scholar with the College of Engineering–Center for Environmental Research and Technology (CE-CERT), University of California at Riverside (UCR), USA. Since 2017, she has been a Post-Doctoral Researcher with the Department of Automation, Tsinghua University. Her current research interests include intelligent transportation systems, mesoscopic energy/emissions modeling, eco-routing, eco-driving, and intelligent vehicle-infrastructure cooperative systems.