

# Decentralized Cooperative Planning for Automated Vehicles with Continuous Monte Carlo Tree Search

Karl Kurzer<sup>1</sup>, Florian Engelhorn<sup>1</sup> and J. Marius Zöllner<sup>1,2</sup>

**Abstract**—Urban traffic scenarios often require a high degree of cooperation between traffic participants to ensure safety and efficiency. Observing the behavior of others, humans infer whether or not others are cooperating. This work aims to extend the capabilities of automated vehicles, enabling them to cooperate implicitly in heterogeneous environments. Continuous actions allow for arbitrary trajectories and hence are applicable to a much wider class of problems than existing cooperative approaches with discrete action spaces. Based on cooperative modeling of other agents, Monte Carlo Tree Search (MCTS) in conjunction with Decoupled-UCT evaluates the action-values of each agent in a cooperative and decentralized way, respecting the interdependence of actions among traffic participants. The extension to continuous action spaces is addressed by incorporating novel MCTS-specific enhancements for efficient search space exploration. The proposed algorithm is evaluated under different scenarios, showing that the algorithm is able to achieve effective cooperative planning and generate solutions egocentric planning fails to identify.

## I. INTRODUCTION

While the capabilities of automated vehicles are evolving, they still lack an essential component that distinguishes them from human drivers in their behavior - the ability to cooperate (implicitly) with others. Unlike today's automated vehicles, human drivers include the (subtle) actions and intentions of other drivers in their decisions. Thus they are able to demand or offer cooperative behavior even without explicit communication. In recent years many research projects have addressed cooperative driving. Yet, the focus to date has been on explicit cooperation, which requires communication between vehicles or vehicles and infrastructure ([1], [2], [3]).

In the foreseeable future neither all vehicles will have the necessary technical equipment to enable communication between vehicles and the infrastructure, nor will algorithms be standardized to such an extent that communicated environmental information and behavioral decisions will be considered uniformly. Hence, automated vehicles should be able to cooperate with other traffic participants even without communication.

Automated vehicles nowadays conduct non-cooperative planning, neglecting the interdependence in decision-making. In general this leads to sub-optimal plans and in the worst case to situations that can only be mitigated by emergency actions such as braking. If the perception and decision making by the individual road user is taken into account,

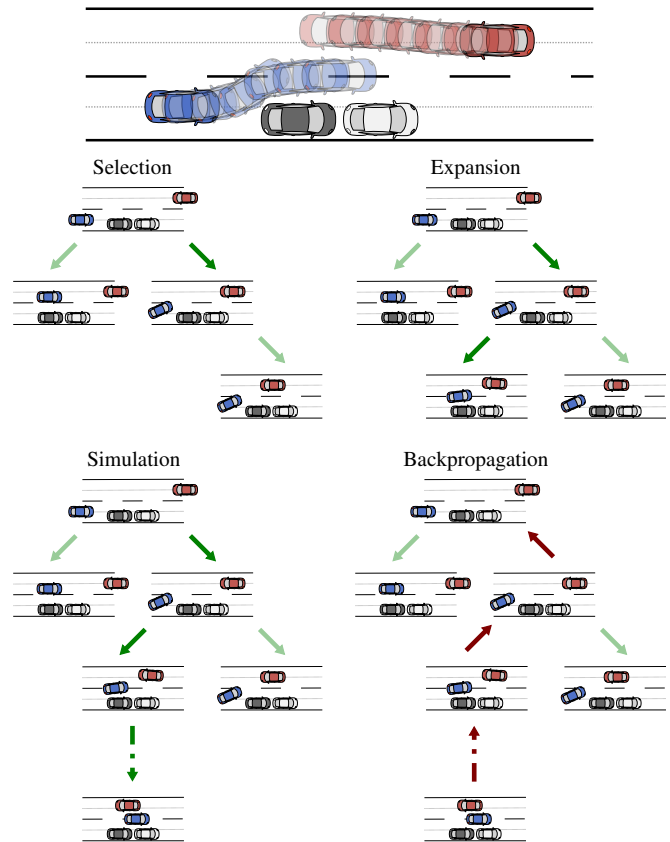


Fig. 1: Phases of Monte Carlo Tree Search for a passing maneuver; the selection phase descends the tree by selecting auspicious future states until a state is encountered that has untried actions left. After the expansion of the state a simulation of subsequent actions is run until the planning horizon is reached. The result is backpropagated to all states along the selected path. Ultimately this process converges to the optimal policy.

trajectory sequences can be planned with foresight and safety-critical traffic situations can be prevented.

Existing frameworks that integrate prediction and planning to model interdependencies and achieve cooperative decision making ([4], [5], [6], [7]) are restricted to a discrete set of actions. However, the multitude of possible traffic scenarios in urban environments require a holistic solution without a priori discretization of the action space.

Thus this work develops a cooperative situation-independent selection of possible actions for each traffic participant, addressing situations with a high degree of interaction between road users, e.g., road constrictions due to parked vehicles or situations that require to merge into moving traffic.

The problem of cooperative trajectory planning is mod-

<sup>1</sup>FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany  
zoellner@fzi.de <sup>2</sup>Karlsruhe Institute of Technology, Kaiserstr. 12, 76131 Karlsruhe, Germany kurzer@kit.edu, florian.engelhorn@gmail.com

eled as a multi-agent markov decision process (MDP) with simultaneous decision making by all agents [8]. With the help of Monte Carlo Tree Search (MCTS) the optimal policy is inferred. Monte Carlo Tree Search, a special reinforcement learning method [9], has shown great potential during multiple occasions facing problems with large branching factors. AlphaGo, the Go software that reached super-human performance is the most prominent example ([10], [11]). MCTS improves value estimates of actions by conducting model-based simulations until a terminal state is reached and uses backpropagation to update the nodes along the taken action sequence. These improved value estimates are essential to direct the selection and expansion phases towards promising areas of the search space. Browne et al. [12] present a thorough overview of MCTS and its extensions. An example for the domain of automated driving is depicted in Fig. 1.

The resulting algorithm, DeCoC-MCTS, plans decentralized cooperative trajectories sampling continuous action spaces. The problem of decentralized simultaneous decision making is modeled as a matrix game and solved by Decoupled-UCT (a variant of MCTS, [13], [7]), removing dependencies on the decisions of others. In order to cope with the combinatorial explosion resulting from continuous action spaces, we enhance the plain MCTS with semantic move groups, kernel updates and guided exploration. The resulting algorithm is able to perform cooperative maneuvers in tight spaces that are common in urban environments.

This paper is structured as follows: First, a brief overview of research on cooperative automated driving is given in section II. Section III introduces the terminology and defines the problem formally. The general approach to the problem and enhancements to the plain MCTS are presented in Section IV. Last, DeCoC-MCTS's extensions for continuous action spaces are evaluated and its capabilities are compared to other planning methods.

## II. RELATED WORK

Cooperative planning takes the actions, intentions and interdependencies of all traffic participants into consideration and seeks to maximize the total utility by following the best combined action. It is important to note that cooperation does not need to be beneficial for each agent (rational cooperation), but that it is sufficient if the combined utility increases given a reference point ([2], [14]).

One approach that generates cooperative plans for highway scenarios first determines the best individual plan using an egoistic driver model. In case this plan results in a conflict a recursive pairwise conflict resolution process is initiated based on the assumption that decisions depend on the traffic ahead. The algorithm performs an exhaustive search over the available maneuver combinations and the lowest non conflicting solution is chosen [15].

Similarly Düring et al. [2] conduct an exhaustive search over a communicated set of discrete actions between two vehicles and choose the joint action with the minimum cost. Building on this, extensions have been developed that aim to

incorporate fairness improvements avoiding that cooperation becomes single sided [14].

Instead of conducting an exhaustive search that quickly becomes intractable for multiple traffic participants and longer time horizons, the problem of cooperative decision making has been solved employing Monte Carlo Tree Search to estimate the best maneuver combination over multiple time steps ([5], [7]).

Other approaches are not explicitly cooperative, however they do capture the interdependencies of actions as they evaluate the threat resulting from different maneuver combinations, and hence predict the future motions of vehicles [4] and are able to generate proactive cooperative driving actions [6].

Furthermore, off-line calculated maneuver templates can be used to devise cooperative plans. First, the maneuver template needs to match a given traffic scene with specific initial constraints, then the maneuver described in the template is checked for feasibility. If multiple templates are feasible given a specific traffic scene the maneuver template with the lowest cost specifies the cooperative maneuver [16].

While the aforementioned approaches do conduct cooperative planning and most of the approaches consider the interdependency between the decisions of the individual traffic participants, some require communication and none are able to plan cooperative maneuvers for arbitrary action spaces that are required by obstructed road layouts and tight urban spaces.

## III. PROBLEM STATEMENT

The problem of cooperative trajectory planning is formulated as a decentralized Markov Decision Process (Dec-MDP). Agents independently choose an action in each time step without knowing the decisions taken by others. Each agent collects an immediate reward and the system is transferred to the next state. Being a cooperative multi-agent system the state transition as well as the reward are dependent on all agents' actions.

The Dec-MDP is described by the tuple  $\langle \Upsilon, \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ , presented in [7].

- $\Upsilon$  denotes the set of *agents* indexed by  $i \in 1, 2, \dots, n$ .
- $\mathcal{S}^i$  denotes the *state space* of an agent,  $\mathcal{S} = \times \mathcal{S}^i$  represents the joint state space of  $\Upsilon$ .
- $\mathcal{A}^i$  denotes the *action space* of an agent,  $\mathcal{A} = \times \mathcal{A}^i$  represents the joint action space of  $\Upsilon$ .
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  is the *transition function*  $P(s'|s, \mathbf{a})$  specifying the probability of a transition from state  $s$  to state  $s'$  given the joint action  $\mathbf{a}$  chosen by each agent independently.
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is the reward function with  $r(s, s', \mathbf{a})$  denoting the resulting reward of the joint action  $\mathbf{a}$ .
- $\gamma \in [0, 1]$  denotes a *discount factor* controlling the influence of future rewards on the current state.

The superscript  $i$  is used to indicate that a parameter relates to a specific agent  $i$ . The joint policy  $\Pi = \langle \pi^1, \dots, \pi^n \rangle$  is a solution to the cooperative decision problem. An individual

policy,  $\pi^i$ , for a single agent, i.e., a mapping from the state to the probability of each available action is given by  $\pi^i : \mathcal{S}^i \times \mathcal{A}^i \rightarrow [0, 1]$ .

It is the aim of each agent to maximize its expected cumulative reward in the MDP, starting from its current state:  $G = \sum \gamma^t r(s, s', a)$  where  $t$  denotes the time and  $G$  the return, representing the cumulated discounted reward.  $V(s)$  is called the state-value function, given by  $V^\pi(s) = E[G|s, \pi]$ . Similarly, the action-value function  $Q(s, a)$  is defined as  $Q^\pi(s, a) = E[G|s, a]$ , representing the expected return of choosing action  $a$  in state  $s$ .

The optimal policy starting from state  $s$  is defined as  $\pi^* = \arg \max_\pi V^\pi(s)$ . The state-value function is optimal under the optimal policy:  $\max V = V^{\pi^*}$ , the same is true for the action-value function:  $\max Q = Q^{\pi^*}$ . The optimal policy is found by maximizing over  $Q^*(s, a)$ :

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathcal{A}} Q^*(s, a) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The optimal policies can easily be derived, once  $Q^*$  has been determined. Hence the goal is to learn the optimal action-value function  $Q^*(s, a)$  for a given state-action combination.

#### IV. APPROACH

While discrete actions as opposed to classical trajectory planning allows to plan over longer periods of time [7], the resolution is insufficient to plan detailed maneuvers. Due to the multitude of possible traffic scenarios in urban environments, a solution with heuristic a priori discretization of the action space of road users is not suitable ([2], [5], [6], [7]). The planning of safe maneuvers within such a dynamic environment can only take place if the trajectory planning is equipped with a situation-independent selection of possible actions for each road user. In order to address driving situations with a high degree of interaction between the road users involved such as road constrictions due to parked vehicles, we employ a MCTS-based approach similar to ([7], [5]) but extend it to continuous action spaces. By extending the method, any desired trajectory sequence can be generated in order to solve complex scenarios through cooperative behavior.

The following subsections first explain the action space and the validation of actions. Based on actions and the resulting state the cooperative reward is described in the following subsection. The last subsections shine some light on the enhancements for continuous action spaces that enable a structured exploration and thus quicker convergence.

##### A. Action Space

Actions are applied to the vehicle's state which is given by its position, velocity and acceleration in longitudinal and lateral direction as well as its heading. An action is defined as a pair of values with a longitudinal velocity change  $\Delta v_{longitudinal}$  and a lateral position change  $\Delta y_{lateral}$ . The value pair describes the desired change of state to be achieved during the action duration  $\Delta T = t_1 - t_0$ . Using this pair of values in combination with the following initial and terminal

conditions for the longitudinal as well as lateral direction, quintic polynomials are solved to generate jerk-minimizing trajectories for the respective direction [17].

The initial conditions for the longitudinal as well as lateral position, velocity and acceleration are determined by the current state. The terminal conditions are defined with:

$$\ddot{x}(t_1) = 0 \quad (2)$$

$$\dot{y}(t_1) = 0 \quad (3)$$

$$\ddot{y}(t_1) = 0 \quad (4)$$

The initial and terminal conditions leave three free constraints with  $\dot{x}(t_1)$ ,  $y(t_1)$  and  $x(t_1)$ . The desired velocity change in longitudinal direction  $\Delta v_{longitudinal}$  (6), as well as the lateral offset of the trajectory  $\Delta y_{lateral}$  (5) are parameterized and introduced as dimensions of the action space of each agent.

$$\dot{x}(t_1) = \dot{x}(t_0) + \Delta v_{longitudinal} \quad (5)$$

$$y(t_1) = y(t_0) + \Delta y_{lateral} \quad (6)$$

The last free unknown is the distance covered in longitudinal direction, (7). In order to describe a physically feasible maneuver, it is defined as

$$x(t_1) = \frac{\dot{x}(t_0) + \dot{x}(t_1)}{2} \Delta T \quad (7)$$

##### B. Action Validation

The resulting trajectory needs to be drivable for a front axle-controlled vehicle and should be directly trackable by a trajectory controller. Hence, we need to conduct an action validation, using kinematic as well as physical boundary conditions so that the simulation adheres to the constraints.

The continuity of the curvature, the steering angle as well as the vehicle dependent permissible minimal radii ensure that the resulting trajectories are drivable. In addition, the dynamic limits of power transmission limit the maximum and minimum acceleration of the vehicle.

##### C. Cooperative Reward Calculation

The immediate individual reward based on the behavior of an agent is calculated using (8). It considers the states reached and the actions taken, as well as a validation reward.

$$r^i = r_{state}^i + r_{action}^i + r_{validation}^i \quad (8)$$

The state reward  $r_{state}^i$  is determined given the divergence between the current and the desired state. A desired state is characterized by a longitudinal velocity  $v_{des}$  and a lane index  $k_{des}$ . To ensure that the agent drives in the middle of a lane, deviations from the lane's center line  $\Delta y$  inflict penalties.

Actions always result in negative rewards (costs). They create a balance between the goal of minimizing the deviation from the desired state as quickly as possible and the most economical way to achieve this. Currently,  $r_{action}^i$  considers only basic properties such as the longitudinal and lateral acceleration (9) and (10) as well as lane changes, (11).

However, they can easily be extended to capture additional safety, efficiency and comfort related aspects of the generated trajectories. The order of importance is adjusted with the respective weights.

$$C_{\ddot{x}} = w_{\ddot{x}} \int_{t_0}^{t_1} (\ddot{x}(t))^2 dt \quad (9)$$

$$C_{\ddot{y}} = w_{\ddot{y}} \int_{t_0}^{t_1} (\ddot{y}(t))^2 dt \quad (10)$$

$$C_{k\pm} = w_{k\pm} \quad (11)$$

The last term is the action validation reward, see (12). It evaluates whether a state and action is valid, i.e., being inside the drivable environment and adhering to the kinematic as well as physical constraints and whether a state action combination is collision free.

$$r_{\text{validation}}^i = r_{\text{invalid state}}^i + r_{\text{invalid action}}^i + r_{\text{collision}}^i \quad (12)$$

To achieve cooperative behavior a cooperative reward  $r_{\text{coop}}^i$  is defined. The cooperative reward of an agent  $i$  is the sum of its own rewards, see (8), as well as the sum of all other rewards of all other agents multiplied by a cooperation factor  $\lambda^i$ , see (13), ([5], [7]). The cooperation factor determines the agent's willingness to cooperate with other agents (from  $\lambda^i = 0$  *egoistic*, to  $\lambda^i = 1$  *fully cooperative*).

$$r_{\text{coop}}^i = r^i + \lambda^i \sum_{j=0, j \neq i}^n r^j \quad (13)$$

#### D. Progressive Widening

In the basic version of the Monte Carlo Tree Search, agents are modeled with discrete action spaces of constant size. Since each action must be visited at least once when using the UCT algorithm [18], the method cannot simply be applied to continuous action spaces. Using *Progressive Unpruning* [19] or *Progressive Widening* [20] the number of discrete actions within the action space can be gradually increased at runtime. A larger number of available actions increases the branching factor of the search tree.

At the beginning, the agent receives an initial, discrete set of available actions in each state. If a state has been visited sufficiently often, it gets progressively widened, by adding another action to the action space. Note that progressive widening is conducted on a per agent basis. The criterion for progressive widening is defined as:

$$N(A(s)) \geq C_{PW} \cdot n(s)^{\alpha_{PW}} \quad (14)$$

The number of possible actions  $N(A(s))$  in state  $s$  therefore depends directly on the visit count of the state  $n(s)$ . The parameters  $C_{PW}$  and  $\alpha_{PW} \in [0, 1]$  must be adapted empirically to the respective application.

The simplest way to add new actions is to select a random action from the continuous action space. More advanced approaches use the information available in the current state, so that promising areas within the action space can be identified and new actions can be added ([21], [22]).

#### E. Semantic Action Grouping

The following section introduces the concept of semantic action grouping. The approach taken is similar to the concepts of ([23], [24]).

The goal of grouping similar actions is to reduce the computational complexity and increase the robustness of the algorithm. The grouping of actions is connected with the implementation of heuristic application knowledge.

Each action is uniquely assigned to a state-dependent action group using a criterion. During the update phase, the characteristic values of the groups are calculated on the basis of the characteristic values of the group members. In the selection strategy, the best action group according to UCT is first selected using the group statistics before a specific action is identified within the group. The action groups serve as filters for determining relevant areas of the action space. Within the action groups, the information of the individual actions is generalized to the group-specific values, but remains unaffected. By using an unambiguous assignment function from the action space to the group action space it is ensured that the action groups do not overlap.

Semantic action groups describe state-dependent, discrete areas within an agent's action space. The areas divide the action space into the following nine areas based on the semantic state description of the agent's future state, as depicted in Fig. 2:

- *No/slight change of state*: 0
- *accelerate*: +
- *decelerate*: −
- *decelerated lane change to the left*: L−
- *lane change to the left*: L
- *accelerated lane change to the left*: L+
- *decelerated lane change to the right*: R−
- *lane change to the right*: R
- *accelerated lane change to the right*: R+

#### F. Similarity Update

As shown in Fig. 3, actions can generate similar ego states independently of the semantic description of their subsequent state and the resulting action group. For example, the difference between the next state of action *I* and *II* is a different track assignment. Although this describes different semantic states, the lateral position deviates only slightly. In contrast, action *III* is assigned to the same action group as action *I*, but has a much larger difference with regard to the lateral deviation. To overcome the restrictions of non-overlapping semantic groups, the values of an agent's actions are generalized to similar actions in the update step, regardless of the action groupings and the resulting boundaries within the action space. During the backpropagation phase

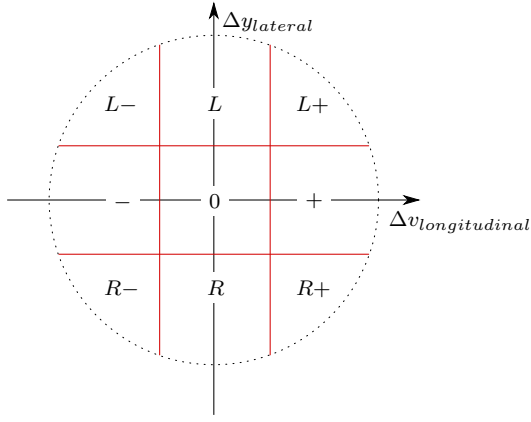


Fig. 2: Segmentation of the action space into semantic action groups; The semantic action groups are dependent on the successor state of the agent's current state, with  $L$  and  $R$  denoting lane changes to the left and right lane and  $+$  and  $-$  acceleration and deceleration respectively.

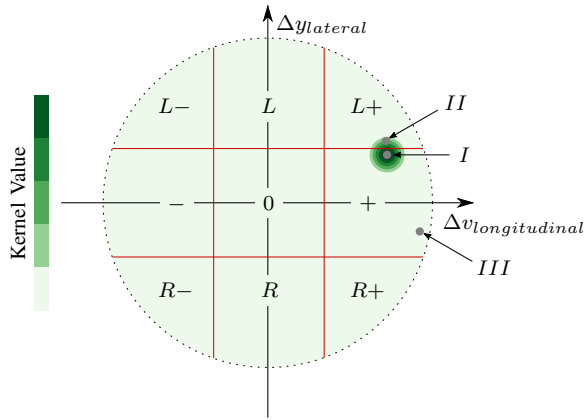


Fig. 3: Similarity update between two actions from different semantic action groups; In the update phase the result from action  $I$  is used to update actions that are within the kernel, i.e.  $II$ , and thus lead to a similar subsequent state.

of MCTS, the similarity between the current action  $a'$  and all previously explored actions of the agent at the expanded state is determined using a distance measure. This measure of similarity is then used to weight the result of the current sample and select similar states to update. The similarity measure is calculated as

$$K(a, a') = \exp(-\gamma \|a - a'\|^2) \forall a \in A_{exp}(s) \quad (15)$$

With the help of the radial basis function a symmetrical distance dimension in the value range  $K(a_i, a^*) \in [0, 1]$  can be mapped. The different dimensions of the action space can also be weighted differently. Since both dimensions are of the same order of magnitude, the same weighting is used for both dimensions. Due to the continuous nature of the kernel the visit count of states is no longer an integer, but a floating point number.

### G. Guided Search

In [21] a heuristic – the so-called *Blind Values* (BV) –  $n$  randomly drawn actions of the theoretical action space

$A$  are evaluated and the action with the resulting maximum Blind Value, see (16), is added to the locally available action space of the state. Blind values make use of the information gathered on all actions already explored from a given state in the past to select promising areas of exploration.

$$a^* = \arg \max_{a' \in A_{rnd}} BV(a', \rho, A_{exp}) \quad (16)$$

The basic idea is to select actions at the beginning, which have a large distance to previous actions in order to foster exploration, avoiding local optima. However, as the visit count increases, actions with a short distance to actions with high UCT values are preferred. To regularize the sum of both criteria, the statistical properties of the previous data points – mean value and standard deviation – are used.

The blind value as a measure of the attractiveness of an action is calculated with

$$BV(a', \rho, A_{exp}) = \min_{a \in A_{exp}} UCT(a) + \rho \cdot K(a, a') \quad (17)$$

where

$$\rho = \frac{\sigma_{a \in A_{exp}}(UCT(a))}{\sigma_{a' \in A_{rnd}}(K(0, a'))} \quad (18)$$

$0$  : center of action space

$K(a, a')$  : distance function for two actions

$A_{exp}$  : set of discrete, previously explored actions

$A_{rnd}$  : set of discrete, randomly selected actions

So that the current statistics of the agent and its action space are being considered.

## V. EVALUATION

The evaluation is conducted using a simulation. The developed algorithm is evaluated under two different scenarios, namely *bottleneck* and *merge-in*, (see Fig. 4). First, the enhancements with regard to continuous actions are evaluated. Second, we demonstrate that our algorithm can achieve effective cooperative planning and generate solutions egocentric planning fails to identify. A video of the algorithm in execution can be found online <sup>1</sup>.

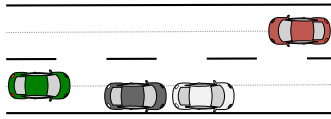
### A. Search Space Exploration

To get a better understanding of the effects of the enhancements and their combinations, the exploration of the agent's action space is evaluated for the root node. This is done using the bottleneck scenario, depicted in Fig. 4a.

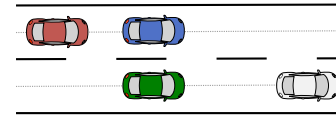
Shortly after the start of the scenario, the green agent changes to the left for all variants of the enhancements. This point is used to analyze the exploration of the action space exemplarily. The actions available to the green agent are shown in Fig. 5. The action with the highest action-value is selected for execution (red triangle).

The basic MCTS algorithm explores the search space randomly, (Fig. 5a). There is no connection between the distribution of samples and the selected action. In comparison,

<sup>1</sup><http://url.fzi.de/DeCoC-MCTS-ITSC>



(a) Bottleneck scenario; gray vehicles are parked, the red and green vehicles can only pass at the same time if the red vehicle cooperates by moving to the right



(b) Merge-in scenario; the gray vehicle is parked, the green vehicle can either, accelerate, passing first, merge-in between the red and the blue vehicle, or decelerate to pass last

Fig. 4: Two evaluation scenarios

the combination with the guided search results in extreme accumulations of actions in the area of an accelerated lane change, (Fig. 5b). Other areas of the action space are neglected. The selected action lies in an unexplored area of the action space.

Using semantic action groups the result clearly differs, (cf. Fig. 5c), as the exploration is more structured. The number of samples in the action space is heavily reduced. Within each semantic action group, actions are randomly distributed. The action groups for lane changes to the left are visited more frequently and one of these action groups entails the final selection.

If semantic action groups are additionally combined with the guided search, (see Fig. 5d), samples accumulate in the area of a lane change to the left. The selected action is close to the highest action density.

Adding the similarity update to the previous configuration a similar behavior can be observed, (Fig. 5e). In addition, there is a strong focus on the target region in form of a high density of possible actions. The selected action is contained in this densely explored area.

The combination of all extensions achieves a structured exploration, requiring less samples due to its higher efficiency than the basic algorithm, indicating the potential of the methods introduced.

### B. Scenarios

Using the scenarios depicted in Fig. 4 DeCoC-MCTS is evaluated for two different predictions. Namely, that other vehicles do not cooperate and keep their velocity (constant velocity assumption) as well as that other vehicles will cooperate. We conduct a qualitative evaluation based on the velocity deviation of all vehicles in a scene, where lower overall deviations indicate more efficient driving and a higher total utility.

The first is the defensive constant velocity assumption that is common in the vast amount of prediction and planning frameworks. This means that the red and blue vehicle will keep their velocity and thus do not cooperate. Fig. 6 and Fig. 7 show the velocity and position graphs for the respective scenario. The green agent  $g_0$  needs to decelerate heavily in order to let the red vehicle pass before it can accelerate again to its desired velocity. A similar behavior with a greater deceleration is required for the merge-in scenario, with the green agent being required to nearly come to a full stop.

If the agent assumes that the other agents do not follow the naive constant velocity assumption, but rather take the perception and decision making by all traffic participants into

consideration, a more efficient cooperative driving style can be observed. The respective velocity and position graphs are depicted in Fig. 8 and Fig. 9. During the bottleneck scenario, both vehicles decelerate only slightly and pass each other at the road constriction. During the merge-in, the blue vehicle accelerates slightly while the red vehicle decelerates to allow the green vehicle to merge in.

It was shown, that the constant velocity assumption can lead to suboptimal solutions, that can be avoided if one takes the interdependence of actions into account, reaching superior solutions with regards to efficiency.

## VI. CONCLUSIONS

This paper proposes a method to plan decentralized cooperative continuous trajectories for multiple agents to increase the efficiency of automated driving especially in urban scenarios and tight spaces. Due to continuous action spaces arbitrary trajectory combinations can be planned, distinguishing this method from other cooperative planning approaches. In order to handle the combinatorial explosion introduced by the continuous action space, we enhanced the MCTS-based algorithm by guided search, semantic action groups and similarity updates. While these enhancements show promising results with regard to the exploration of the search space, current research focuses on tuning the methods, proving scenario independent applicability and conduct in depth quantitative analysis.

## VII. ACKNOWLEDGEMENTS

We wish to thank the German Research Foundation (DFG) for funding the project Cooperatively Interacting Automobiles (CoInCar) within which the research leading to this contribution was conducted. The information as well as views presented in this publication are solely the ones expressed by the authors.



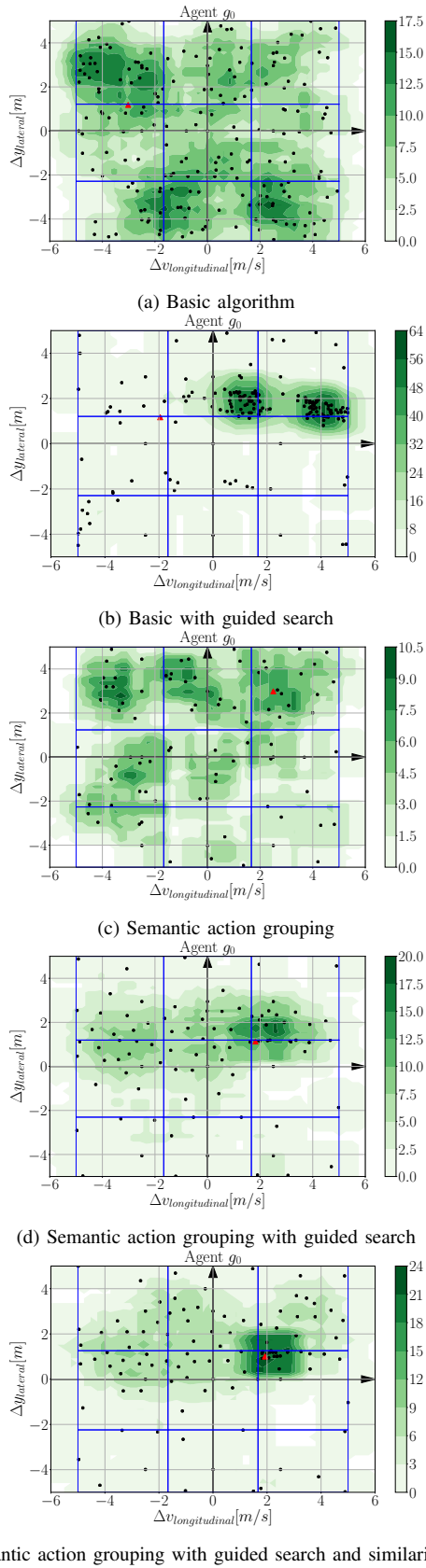


Fig. 5: Comparison of the search space exploration of the green agent during a lane change in the bottleneck scenario, Fig. 4a. Each sample marks an explored action. The color scale shows the sample density. The state limits introduced by the semantic action groups are shown in blue. The red triangle marks the selected action. The selection is made on the basis of the highest action-value.

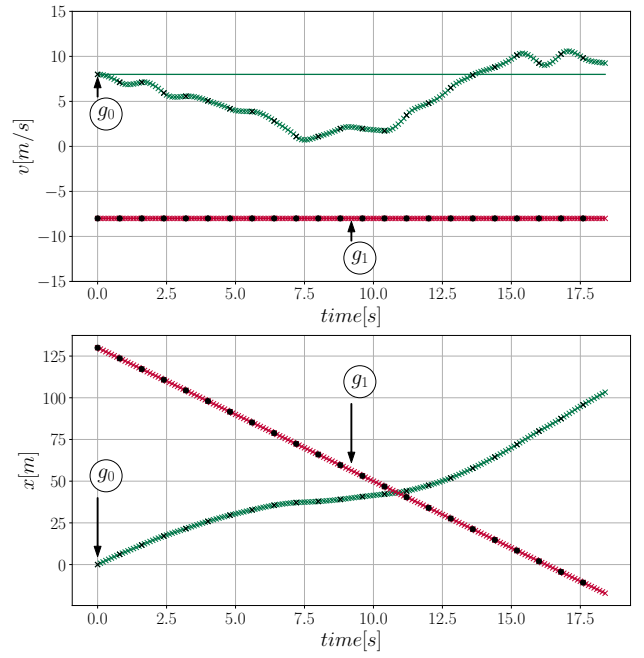


Fig. 6: Bottleneck scenario with constant velocity assumption; agent  $g_0$  reduces its velocity heavily, while agent  $g_1$  approaches the obstacle with its desired velocity.

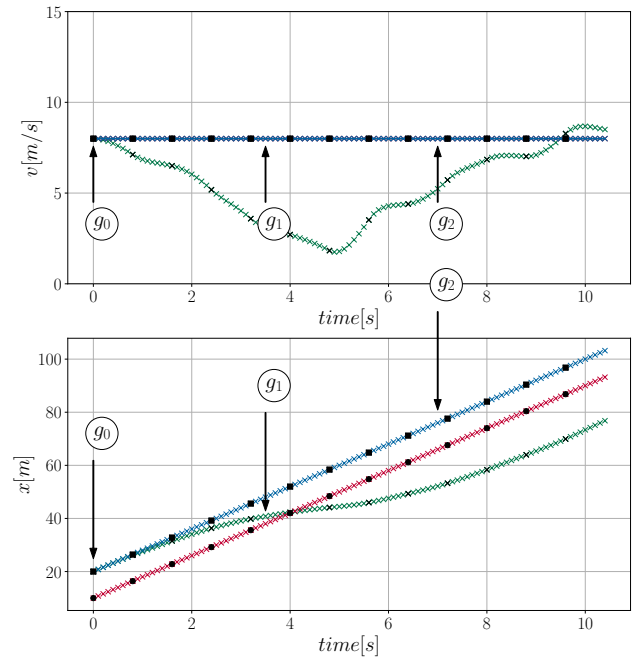


Fig. 7: Merge-in scenario with constant velocity assumption; agent  $g_0$  reduces its velocity heavily, while agent  $g_1$  and  $g_2$  continue with their desired velocity

## REFERENCES

- [1] C. Englund *et al.*, “The Grand Cooperative Driving Challenge 2016: Boosting the introduction of cooperative automated vehicles,” *IEEE Wireless Communications*, vol. 23, 2016.
- [2] M. Düring *et al.*, “Cooperative decentralized decision making for conflict resolution among autonomous agents,” in *International Conference on Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2014.
- [3] C. Frese, J. Beyerer, and P. Zimmer, “Cooperation of Cars and Formation of Cooperative Groups,” in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2007.
- [4] A. Lawitzky, D. Althoff, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, “Interactive scene prediction for automotive applications,” in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013.
- [5] D. Lenz *et al.*, “Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search,” in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016.
- [6] M. Bahram *et al.*, “A Game-Theoretic Approach to Replanning-Aware Interactive Scene Prediction and Planning,” *IEEE Transactions on Vehicular Technology*, vol. 65, 2016.
- [7] K. Kurzer *et al.*, “Decentralized Cooperative Planning for Automated Vehicles with Hierarchical Monte Carlo Tree Search,” in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, in press.
- [8] P. I. Cowling, E. J. Powley, and D. Whitehouse, “Information Set Monte Carlo Tree Search,” *IEEE Transactions on Computational Intelligence and AI in Games (T-CIAIG)*, vol. 4, 2012.
- [9] T. Vodopivec, S. Samothrakakis, and B. Ster, “On monte carlo tree search and reinforcement learning,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 60, 2017.
- [10] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, 2016.
- [11] —, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, 2017.
- [12] C. B. Browne, E. Powley, *et al.*, “A survey of Monte Carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games (T-CIAIG)*, vol. 4, 2012.
- [13] M. J. W. Tak, M. Lanctot, and M. H. M. Winands, “Monte carlo tree search variants for simultaneous move games,” in *IEEE Computational Intelligence and Games (CIG)*, 2014.
- [14] “Advanced cooperative decentralized decision making using a cooperative reward system,” in *International Conference on Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2015.
- [15] W. Schwarting and P. Pascheka, “Recursive conflict resolution for cooperative motion planning in dynamic highway traffic,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014.
- [16] S. Manzingier, M. Leibold, and M. Althoff, “Driving strategy selection for cooperative vehicles using maneuver templates,” in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017.
- [17] A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto, “Local path planning and motion control for agv in positioning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 1989.
- [18] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *European Conference on Machine Learning (ECML)*, 2006.
- [19] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. v. D. Herik, J. W. H. M. Uiterwijk, and B. Bouzy, “Progressive strategies for monte-carlo tree search,” *New Mathematics and Natural Computation*, vol. 04, 2008.
- [20] R. Coulom, “Computing elo ratings of move patterns in the game of go,” in *Computer Games Workshop (CGW)*, 2007.
- [21] A. Couëtoux, Adrien, H. Doghmen, and O. Teytaud, “Improving the exploration in upper confidence trees,” in *International Conference on Learning and Intelligent Optimization (LION)*. Springer, 2012.
- [22] T. Yee, V. Lisy, and M. Bowling, “Monte carlo tree search in continuous action spaces with execution uncertainty,” in *International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2016.
- [23] J. Takeshi Saito, M. H. M. Win, J. W. H. M. Uiterwijk, and H. J. V. D. Herik, “Grouping Nodes for Monte-Carlo Tree Search,” in *Computer Games Workshop (CGW)*, 2007.
- [24] B. E. Childs, J. H. Brodeur, and L. Kocsis, “Transpositions and Move Groups in Monte Carlo Tree Search,” in *IEEE Computational Intelligence and Games (CIG)*. IEEE, 2008.

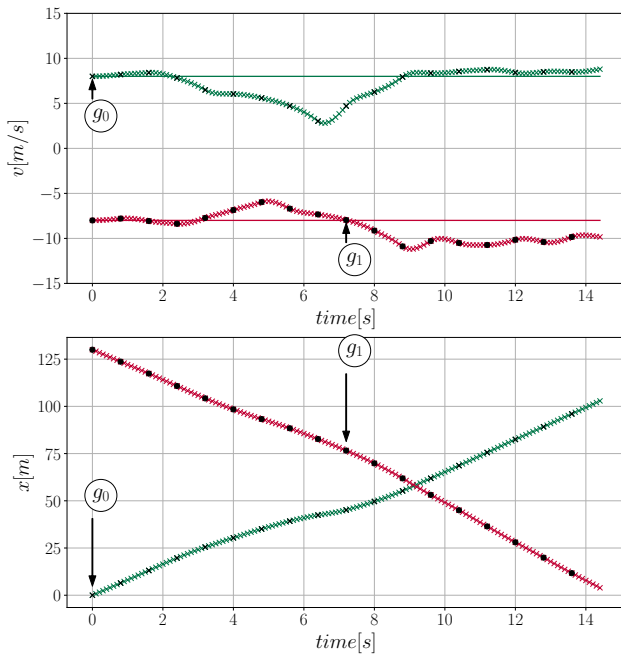


Fig. 8: Bottleneck scenario with the assumption of cooperative behavior; agent  $g_0$  and  $g_1$  reduce their velocity slightly

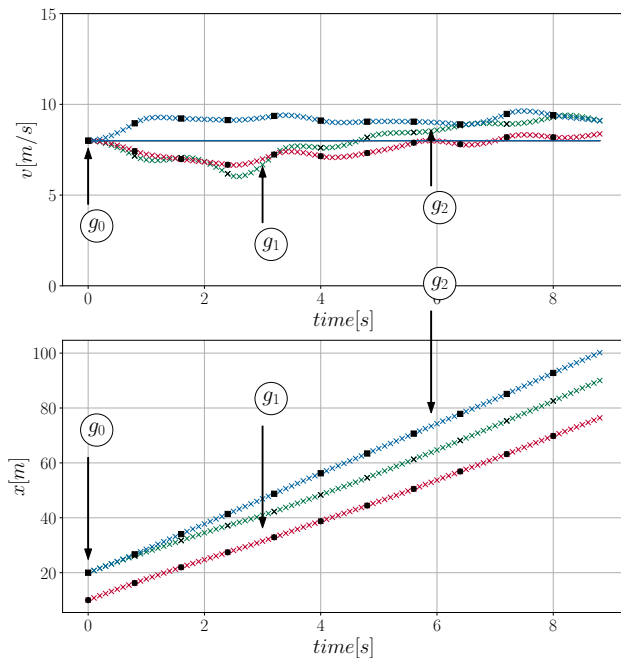


Fig. 9: Merge-in scenario with the assumption of cooperative behavior; agent  $g_2$  accelerates slightly while agent  $g_1$  decelerates slightly to open a gap so that agent  $g_0$  can merge-in with little deceleration