2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

Adaptive Reward for CAV Action Planning using Monte Carlo Tree Search

Dhruvkumar Patel[1] and Rym Zalila-Wenkstern[2]

***Abstract*— Cooperative action planning for Connected and Autonomous Vehicles (CAVs) in an emergency scenario is an important task in the autonomous driving domain. Reinforcement learning algorithms such as Monte Carlo Tree Search (MCTS) have popularly been used to solve this problem with some success. MCTS rely on performing many simulations of CAV actions to learn expected reward values for CAV actions. A refined reward function design is a necessary precondition for better success rates in MCTS. Traditionally, predefined reward functions with fixed reward parameters are used in all CAVs scenarios by most MCTS-based algorithms. This paper presents a novel Monte Carlo Tree Search (MCTS) based algorithm that dynamically modifies the reward function parameters to encourage or discourage particular CAV actions. Our proposed algorithm with a dynamic reward function significantly improves the reliability of MCTS having a fixed reward function. We evaluate the proposed algorithm in a large-scale multi-agent-based traffic simulation system. Experimental results show that our algorithm significantly improves upon current state-of-the-art centralized and decentralized algorithms.**

I. Introduction

Connected and Autonomous Vehicles (CAVs) in a coalition continuously exchange information and cooperate to make rewarding decisions for the coalition [1]. In case a misbehaving vehicle is detected by one of the coalition's CAVs, the vehicle alerts all coalition members about the emergency. The coalition members must cooperatively plan and find non-conflicting actions to avoid collisions with the misbehaving vehicle when possible.

CAV cooperative action planning approaches generally fall into three main categories: centralized, decentralized, and hierarchical. In centralized approaches, each CAV sends its state information to a central computer. The central computer solves the cooperative action planning problem formulated as a global optimization problem. Centralized approaches generally find optimal solutions but scale poorly when the number of CAVs becomes large [2].

Decentralized approaches formulate the CAV cooperative action planning problem as a Multi-agent Markov Decision Process (MMDP). Given that a CAV's state space can be very large [3], classical reinforcement learning techniques such as value iteration do not provide efficient solutions. A game theory-based technique called Monte Carlo Tree Search (MCTS) [4] has been successful at addressing problems with large state spaces such as the traditional Chinese game, Go [5]. MCTS has also been applied to solve MMDPs for the CAV action planning problem [6]–[8] to some success. Although MCTS-based approaches can tackle problems with large state-spaces, they scale poorly in the number of CAVs. This is due to the fact that the MCTS tree size grows exponentially with the number of CAVs.

In [9], we proposed Approximate Simultaneous Move (ASM), an MCTS-based algorithm that scales with the number of CAVs. ASM follows a hierarchical, two-step approach. First, each CAV computes its prioritized actions in a decentralized manner. Then the final action selection for CAVs is performed by the coalition leader. ASM increases the scalability significantly compared to other MCTS-based approaches.

MCTS-based approaches rely on Monte Carlo simulations of CAV actions to learn the expected reward value. During an MCTS simulation of CAV actions, a predefined reward function is used to compute the reward value of simulating a particular CAV action. The design of the reward function and selection of reward function parameters significantly affect the MCTS algorithm's ability to find a non-colliding solution in a given CAV situation. However, little attention has been granted to the reward function design in selecting reward parameters for different CAV situations.

In this paper, we present **A**daptIve **R**eward for **CA**Vs **A**ction **P**lanning (AirCap), an MCTS-based algorithm that changes the reward function weights dynamically during MCTS iteration simulations. To achieve this, we propose to update the reward function weights for each action, based on the resultant coalition state of that action. AirCap is able to significantly improve upon ASM [9], which uses fixed reward function weights, and performs much better than the state-of-the-art centralized and decentralized algorithms.

This paper is organized as follows: In the next section, we review related works. In Sections III and IV, we formulate the problem and discuss the approach. In Section V, we present the AirCap algorithm and in Section VI, we present a comparative analysis of AirCap with ASM and other state-of-the-art centralized and decentralized algorithms.

II. Related works

Decision making approaches for the CAVs action planning problem can be classified as *centralized*, *decentralized* and *hierarchical*. In centralized approaches, the action planning problem for a group of CAVs is formulated as a single global optimization problem and is solved by a centralized computer. In decentralized and hierarchical approaches, CAVs action planning problem is generally formulated as a form of

[1]Dhruvkumar Patel is with Department of Computer Science, University of Texas at Dallas, 800 W Campbell Rd, Richardson, Texas 75080. dhruv@utdallas.edu

[2]Rym Zalila-Wenkstern is with Department of Computer Science, University of Texas at Dallas, 800 W Campbell Rd, Richardson, Texas 75080. rymw@utdallas.edu

Multi-agent Markov Decision Process (MMDP) and is solved by each CAV independently. In hierarchical approaches, there is an additional step where a CAV coalition leader checks each CAV's individual solutions and selects the best solution for the coalition.

A. Centralized approaches

Most conventional optimization-based approaches for CAVs action planning fall in this category. The proposed solutions differ in their optimization method's selection and definition. [10] uses Mixed Integer Linear Programming (MILP) with the collision constraints. [11] uses Mixed Integer Quadratic Programming (MIQP) with the collision constraints. [12] uses a graph theory based optimization approach. [13] uses pre-defined maneuver templates for collaborative collision avoidance. [14] formulates the problem as a centralized Quadratic Programming (QP) problem, which is decomposed using alternating direction of multipliers. [15] proposes to formulate the trajectory generation problem as an integer programming problem. The integer programming problem is solved using a SAT solver, which provides the formal guarantee for collision avoidance. Two methods are introduced to reduce the computation time. In the grouping method, vehicles are grouped into disjoint sets of interacting vehicles and the optimization is performed for each set separately. In the collision check point method, the optimization is carried out less frequently. [15] uses state-of-the-art centralized optimization approach as the computation time reduction techniques reduce the computation time by 98%.

Centralized approaches generally provide optimal solutions to the formulated optimization problems, however they are computationally expensive and not scalable.

B. Decentralized approaches

In decentralized approaches, the CAVs action planning problem is commonly formulated as a Multi-agent Markov Decision Process (MMDP) and solved using reinforcement learning methods. In this category, each CAV uses an individual reward function for the proposed reinforcement learning approach. Additionally, each CAV has a choice when it comes to the level of cooperation with other CAVs. In [6], authors propose a Monte Carlo Tree Search (MCTS) based approach with an individual multi-objective reward function. Each CAV constructs a cooperative reward function that specifies the level of cooperation with other CAVs and defines the optimized utility. The approach was evaluated on scenarios involving up to three cooperative vehicles. [7] proposes an MCTS-based approach with an individual multi-objective reward function and CAV macro-actions. A similar approach for cooperative reward function is used to define the optimized utility. Additionally, reward shaping is used in the reward function that pushes the CAV state towards desired velocity and desired lane index. The evaluation is performed against scenarios involving up to three vehicles. [8] proposes DeCoC-MCTS, a Decentralized Cooperative Continuous MCTS-based approach with an individual multi-objective reward function. This method proposes to use continuous action-spaces for CAVs to generate flexible trajectories. Cooperative reward function is constructed by each CAV to define the optimized utility. The evaluation is performed against scenarios involving up to three vehicles.

The main drawback of decentralized approaches that have individual reward with social choice utility is that the collision avoidance is only guaranteed when all CAVs reach Nash equilibirium, i.e., they all find the same solution. Another drawback is scalability in the number of agents. [8] is currently considered the state-of-the-art in decentralized CAV action planning. It is the only MCTS-based approach to have used continuous action spaces for CAV actions, allowing flexible trajectories for CAVs.

C. Hierarchical approach

In [9], we have proposed Approximate Simultaneous Move (ASM), an MCTS-based algorithm that reduces the MCTS tree size exponentially to improve scalability . Each CAV first executes ASM to define individual, prioritized action plans up to the planning horizon. Each CAV then sends its prioritized action plans to the coalition leader. The coalition leader chooses the final action plan for each CAV that optimizes the team utility. In ASM, each CAV uses an individual reward function to rank its actions, but the final action selection is performed by the coalition leader to optimize the team utility. All previously proposed decentralized and hierarchical approaches use a fixed weights multi-objective reward function in all coalition configurations. In this paper, we present **A**dapt**I**ve **R**eward for **C**AVs **A**ction **P**lanning (AirCap), a novel MCTS-based algorithm with an adaptive reward function. Each CAV uses different reward function parameters for actions that lead to different coalition states. The unique contributions of our approach are the following:

- AirCap is based on a novel idea of updating the weights of the CAVs reward functions dynamically.
- AirCap was implemented and evaluated in MATISSE, a large scale multi-agent based traffic simulation system.
- Extensive experiments show that AirCap outperforms ASM [9], as well as the state-of-the-art centralized [15] and decentralized [8] algorithms.

III. Model Definition

We use the terms agent and CAV interchangeably in the remainder of this paper. We assume that a coalition $\mathcal{C}$ of n CAVs is navigating on a straight highway. A CAV coalition is a group of CAVs that drive together for information sharing and mutual support. In the coalition, one CAV is chosen as a coalition leader and is responsible for coalition management. We adopt coalition formation and leader assignment algorithm from our previous work [1]. Based on current CAV technologies, we assume that each CAV is equipped with perception sensors such as ultrasonic sensors, radars and Lidar which provide short (i.e., less than 2 meters) and long range (i.e., up to 200 meters) sensing data. CAVs in a coalition use V2V communication to continuously exchange information with each other. The information exchanged depends on the CAV role in the coalition (i.e., members or a leader).

Each CAV in the coalition is defined by its ID $i \in \mathcal{C}$. A CAV i's state is defined by a state vector $s^i = [p^i, v^i, l^i, \theta^i, x^i]$ where p^i is the CAV's position, v^i is its velocity, l^i is its lane, θ^i is its orientation, and x^i is the size of the vehicle containing length and width values. The coalition joint state is defined as $s = \{s^i\}_{i \in \mathcal{C}}$. Table I lists the individual CAV action set. A *mitigation action plan* α_h^i is a sequence of h consecutive actions that is defined by a CAV i in case of perceived danger or warning related to a misbehaving vehicle. It is defined as $\alpha_h^i = \{a_{t_k}^i\}_{k=1}^h$, where t_k is the start time of the execution of the mitigation plan's k'th action. h is known as the *planning horizon*. An action is performed over a duration denoted Δt.

Action	Definition	Precondition
$maintain$	Maintain the current velocity	-
$accel$	Accelerate with a fixed acceleration value α_{acc}	Speed shouldn't exceed maximum speed
$decel$	Decelerate with a fixed deceleration value α_{dec}	Speed shouldn't be zero
cll	Change lane to the left lane of the current lane	Currently not in the left most lane
clr	Change lane to the right lane of the current lane	Currently not in the right most lane

TABLE I: CAV Actions

We formulate the CAVs cooperative action planning problem in an emergency situation as a Multi-agent Markov Decision Process (MMDP) [16]. When a misbehaving vehicle is detected by one of the CAVs, each CAV in the coalition formulates its action planning problem as an MMDP with its individual reward function to compute individual prioritized mitigation action plans. The coalition leader uses these individual action plans to optimize the team utility. An MMDP for the CAV action planning problem is defined by a tuple $\langle \mathcal{C}, S, A, T, R \rangle$, where

- $\mathcal{C}$ is a coalition of n CAVs.
- S represents the joint state space for the CAVs in $\mathcal{C}$. $S = \times S^i$ where S^i is the state space for CAV i.
- A represents the joint action space for the actions of CAVs in $\mathcal{C}$. $A = \times A^i$ where A^i is the set of actions that i can perform.
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function where $T(s, a, s')$ specifies the probability of the system transitioning to state s' when performing joint action a in state s.
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function with $R(s, a, s')$ specifying the reward received when executing joint action a in state s and transitioning to the state s'.

The solution to an MMDP is specified by state-action values $Q(s^i, a^i)$ for each state-action pair. $Q(s^i, a^i)$ represents the expected reward if CAV i performs the action a^i in the state s^i.

IV. Approach

In our approach, in the event that a misbehaving vehicle m is detected, each CAV i independently solves MMDP using the communicated joint state s to derive all possible mitigation action plans for horizon h and estimate approximately optimal Q^* values for the actions in derived action plans. Each CAV then ranks these plans using the estimated Q^* values. Each CAV i sends its ranked plans to the coalition leader using V2V communication. The coalition leader analyzes the prioritized mitigation action plans, and selects an action plan α_h^i for each CAV i that optimizes the team utility. We propose an algorithm based on Monte Carlo Tree Search (MCTS) to solve an MMDP at individual CAV level.

A. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a tree-search based reinforcement learning algorithm used to solve an MDP. It performs several Monte Carlo simulations and uses the simulation outcomes to estimate optimal state-action values Q^*. An MCTS tree consists of nodes referring to MDP states, and edges referring to MDP actions. One iteration of MCTS algorithm involves four stages: *selection*, *expansion*, *simulation* and *backpropagation*. In the selection step, MCTS nodes are selected using a specific selection strategy starting from the root node until a leaf node is reached. In the expansion step, the leaf node is expanded by one or more remaining children nodes, and one of the expanded children nodes is again selected using the selection strategy. In the simulation step, a Monte Carlo simulation is performed from the current node until the planning horizon or until a terminal node is found. In the backpropagation step, the reward outcome of the simulation is used to update each visited node's statistics in a reverse manner from leaf to root.

B. Approximate Simultaneous Move MCTS

In our previous work [9], we proposed Approximate Simultaneous Move (ASM), an MCTS-based algorithm for CAV action planning in the presence of a misbehaving vehicle. ASM is suitable for larger number of agents or action-space sizes than regular MCTS. ASM reduces the MCTS tree size exponentially by reducing the branching factor. Each node in the MCTS tree refers to a coalition joint state. However, each edge in the MCTS tree refers to an individual action of the current agent and not the joint action as in regular MCTS. Since each edge only refers to an individual action for the current agent, the branching factor is reduced by a factor equal to the number of agents. Resultant MCTS tree is exponentially smaller and requires significantly less iterations to explore.

The reward function used in ASM algorithm consists of three reward parameters and is defined as:

$$r^i = r^i_{collision} + r^i_{speed} + r^i_{success} \tag{1}$$

where,

- $r^i_{collision}$ is a fixed negative reward parameter if the simulated action leads to a collision for CAV i.

- $r^i_{success}$ is the fixed positive reward parameter if the selected action for CAV i is simulated successfully without any collisions.
- r^i_{speed} is the sum of two values: the speed deviation for CAV i and a fixed negative reward parameter indicating if CAV i came to a full stop.

C. Limitations of ASM reward function

The most important drawback of the current ASM reward function is that it uses common set of fixed reward parameters for all CAV situations. Based on the initial coalition state and actions selected during MCTS iteration simulations, different coalition states will be simulated during different MCTS iterations. From a coalition state s_{t_k} at time t_k, different CAV actions $a^i_{t_k}, \forall i \in \mathcal{C}$ will lead to different coalition states $s_{t_{k+1}}$ at time t_{k+1}. Using same set of fixed reward parameters for different coalition states is suboptimal. Additionally, the current ASM reward function only considers three reward parameters. These parameters are not representative enough to incentivize or deter different CAV actions that lead to different coalition states. Since some coalition states are more desirable than others for successful collision avoidance, additional reward parameters need to be added to the ASM reward function.

V. AirCap Algorithm

We present **A**dapt**I**ve **R**eward for **CA**V **A**ction **P**lanning(AirCap) algorithm for CAV action planning in an emergency situation. We consider CAVs driving in a coalition formation on a straight highway. A coalition member detects a misbehaving vehicle in its neighborhood and alerts all coalition members. A CAV i's neighborhood is defined by a bounding box centered at $s^i.p^i$ and having a fixed length of 3 times CAV i's length $s^i.x^i.length$ and a fixed width of 2 times CAV i's width $s^i.x^i.width$ as shown in Figure 1. After receiving an alert about the misbehaving

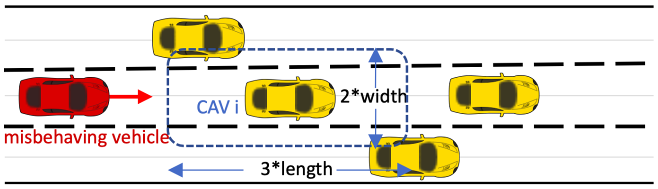


Fig. 1: Neighborhood CAV i

vehicle, each CAV executes the AirCap MCTS algorithm to rank its possible action plans α^i_h for the planning horizon h. Each of these ranked action plans α^i_h consists of a sequence of actions $a^i_{t_k}$ to be performed and their associated Q^* state-action values. To estimate the Q^* values, CAV i simulates many MCTS iterations with a different possible action plan in each iteration. The action plan to be simulated in a particular MCTS iteration is chosen by MCTS selection, expansion and simulation strategies. The reward for taking each of the actions in the simulated action plan is computed using the novel reward function described below.

A. AirCap reward function

We consider several reward factors that affect CAV's safety, efficiency and comfort. Individual CAV i's reward $r^i_{t_k}$ at time t_k is defined as a dot product of two vectors: a weight vector $\overline{w}$ and a reward factors vector $\overline{r}^i$.

$$r^i_{t_k} = \overline{w} \cdot \overline{r}^i \tag{2}$$

which is equivalent to,

$$r^i_{t_k} = \sum_j w_j r^i_j$$

where $\overline{r}^i = \{r^i_j\}$ is a vector consisting of several reward factors indexed by j as follows:

- r^i_{ccol}: *coalition collision* corresponds to the boolean value indicating if there is a collision between CAV i and another coalition member while simulating an action $a^i_{t_k}$ during the current MCTS iteration
- r^i_{mcol}: *misbehaving collision* corresponds to the boolean value indicating if there is a collision between CAV i and the misbehaving vehicle while simulating an action $a^i_{t_k}$ during the current MCTS iteration
- r^i_{accel}: Current action $a^i_{t_k}$'s acceleration
- r^i_{decel}: Current action $a^i_{t_k}$'s deceleration
- r^i_{sd}: *speed deviation* corresponds to the difference of the current speed $s^i_{t_k}.speed$ and the initial speed $s^i_{t_1}.speed$ of the CAV
- r^i_{lc}: *lane change* corresponds to the difference between the current lane index $s^i_{t_k}.lane$ and the previous lane index $s^i_{t_{k-1}}.lane$
- r^i_{stop}: *full stop* corresponds to the boolean value indicating if the current speed $s^i_{t_k}.speed$ equals zero
- r^i_{mv}: *max velocity* corresponds to the boolean value indicating if the current speed $s^i_{t_k}.speed$ goes above the allowed maximum velocity of the vehicle

In Equation 2, the weight vector $\overline{w}$ consists of weight values $\{w_j\}$, each of which is multiplied with exactly one reward factor from the set of reward factors. Each of the reward factors represents one or more action. For example, r^i_{accel} represents *accel* action and r^i_{lc} represents *cll* and *clr* actions. Setting a high or low value for a particular weight w_j makes CAV i to either prioritize or deprioritize that reward factor's represented actions in its final list of prioritized action plans. For example, if we set a relatively high weight value w_{accel} which is multiplied with the reward factor r^i_{accel}, then CAV i prioritizes action *accel* higher than other actions in its final action plans. Since *coalition collisions* are much more frequent than *misbehaving collisions* during AirCap iterations, we split them in separate factors.

An important observation is that some actions should be prioritized in a particular coalition state but in a completely different coalition state, the same of set of actions may lead to a collision among coalition members. Since action plans for different MCTS iterations are generated using Monte Carlo samples and are different from each other, coalition states will also be different for each action in the action plan for each MCTS iteration. Using the same set of weights for all actions in all iterations is suboptimal. Below we discuss the coalition state-based weights updating algorithm for the AirCap reward function.

1) Coalition state based weights update: In AirCap, each CAV i executes MCTS iterations to estimate Q^* values for its explored action plans α_h^i. When simulating a single MCTS iteration, CAV i executes a sequence of actions $\{a_{t_k}^i\}_{k=1}^h$ in the action plan α_h^i. When simulating each of the actions in $\{a_{t_k}^i\}_{k=1}^h$ sequence, coalition states $s_{t_k}^i$ will be different for different k values. For instance, consider the following coalition state at time step t_k for $k \in \{1, h\}$: Consider the

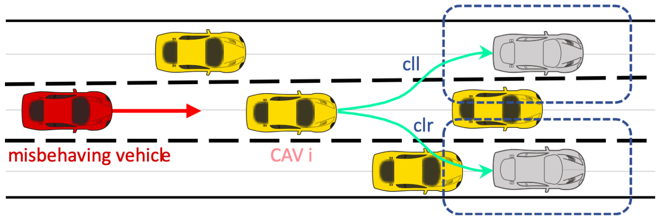


Fig. 2: *cll* and *clr* actions for CAV i at time step t_k

two actions for CAV i at time step t_k: *change lane to left (cll)* and *change lane to right (clr)*. The positions of CAV i at time step t_{k+1} for both actions are shown in figure 2. We consider the neighborhood of each of these positions. There is one coalition member in the neighborhood for action *cll*'s ending position, whereas there are two coalition members in the neighborhood for action *clr*'s ending position. While CAV i selects and simulates its chosen action during current MCTS iteration, it will also select and simulate actions for the neighborhood vehicles simultaneously in the same MCTS iteration. This may lead to a collision with CAV i if the the selected actions of the neighborhood vehicles are conflicting with CAV i's selected action. If we consider all future actions that can be selected for neighborhood vehicles and take all the action combinations of neighborhood vehicles, then the proportion of neighborhood vehicles' action combinations that can have a collision with CAV i are greater if CAV i selects action *clr* than if CAV i selects action *cll*'s ending position. Thus more penalty (or less reward) should be received for choosing action *clr* than action *cll*. We achieve this by updating the reward weights dynamically before simulating each action during the AirCap's MCTS iteration. We assume that the weights are negative values and represent the penalty. Reward weights are temporarily modified for each action $a_{t_k}^i$ as below:

$$\overline{w} = (1 + \beta \mathcal{N}^i)\overline{w} \tag{3}$$

where

- $\mathcal{N}^i$ is the number of vehicles in CAV i's neighborhood after simulating action $a_{t_k}^i$ during current MCTS iteration
- β is the coefficient in range $[0, 1]$

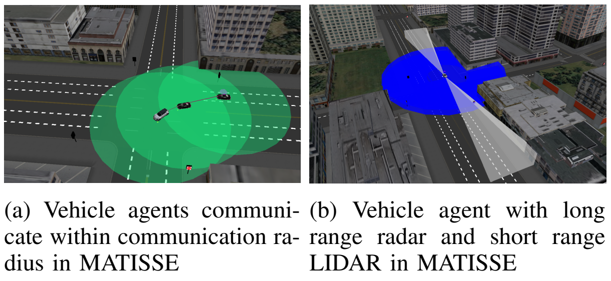
During the simulation of each MCTS iteration in AirCap, CAV i uses the modified reward weights as given in Equation 3 for each action in the simulated action plan until the planning horizon. During the backpropagation phase of MCTS iteration, reward values computed using Equation 2 are used to update each visited node's statistics in the MCTS tree in a reverse order from the leaf node to the root node. After all MCTS iterations are executed, CAV i computes the state-action value Q^* at each node of its MCTS tree using the node statistics. Each CAV i uses the computed Q^* values to construct, rank and share its mitigation action plans with the coalition leader, which finds the best possible non-colliding action plan for the whole coalition using the beam-search algorithm [9].

VI. Experimental Results

In this section, we evaluate the success rate of the proposed AirCap algorithm and compare it with the ASM algorithm [9], the state-of-the-art centralized algorithm proposed by [15] and the leading decentralized algorithm DeCoC-MCTS proposed by [8]. We first discuss the multi-agent based simulation framework used to run the experiments, the experimental setting and then present our results.

A. MATISSE

We tested the algorithms in MATISSE 3.0, a microscopic multi-agent based traffic simulation system developed at the MAVS lab at UT Dallas [17]. MATISSE includes necessary tools and features to simulate for Intelligent Transportation Systems including virtual CAV agents, virtual traffic light agents and virtual intersection agents.



(a) Vehicle agents communicate within communication radius in MATISSE (b) Vehicle agent with long range radar and short range LIDAR in MATISSE

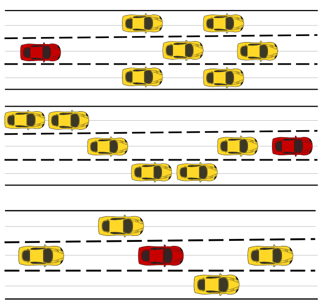
Fig. 3: Realistic simulation of CAVs in MATISSE

A virtual CAV agent features simulated sensors including short range LIDAR and long range radars. CAVs can also communicate with other CAVs using simulated V2V communication. During the simulation, the user can modify a CAV's properties (e.g., sensor range and field of view, communication radius) and behavior (e.g., speed, acceleration) and witness the outcome in simulated real-time.

B. Experimental setting

We have implemented and tested AirCap, ASM, DeCoC-MCTS, and the centralized algorithm in MATISSE, the multi-agent-based traffic simulator [1]. We refer to a unit of length in MATISSE simulator as *unit*. One full simulation cycle of MATISSE is referred as *cycle*. In our experiments, a coalition of CAVs is navigating on a three-lane straight highway at the speed of 2 units/cycle. One of the coalition

[1]Demo videos are available at `https://www.utdmavs.org/connected-vehicles/`


(a) Scenario 1: Misbehaving vehicle accelerates from behind the coalition of 6 CAVs

(b) Scenario 2: Misbehaving vehicle stops in front of the coalition of 6 CAVs

(c) Scenario 3: Misbehaving vehicle accelerates or stops in the middle of the coalition of 4 CAVs

Fig. 4: Placement of coalition members (yellow) and the misbehaving vehicle (red) in tested scenarios

members detects a misbehaving vehicle in its sensor range. We consider three types of scenarios with respect to the misbehaving vehicle's position. Figure 4 illustrates all three scenarios used in our experiments.

- *Scenario 1*: The misbehaving vehicle is positioned behind the coalition and speeds up at 5 units/cycle.
- *Scenario 2*: The misbehaving vehicle is positioned in front of the coalition and comes to a full stop at 0 units/cycle.
- *Scenario 3*: The misbehaving vehicle is surrounded by the coalition members and either speeds up at 5 units/cycle or comes to a full stop at 0 units/cycle.

The affected CAV detects a potential collision with the misbehaving vehicle that will occur after *Time To Collide* (TTC) cycles. After the affected CAV sends alert to other coalition members, each coalition member executes the decision-making algorithm to find its collision-avoiding actions or the collision-avoiding trajectory (depending on the algorithm) for a planning horizon h. The TTC value in these experiments is set to 15 cycles. We set the planning horizon $h = 60$ cycles, significantly greater than TTC, to avoid possible and yet undetected collisions of other coalition members with the misbehaving vehicle. For AirCap algorithm, the reward coefficient β is set to 0.5.

For AirCap, ASM, and DeCoC-MCTS, CAVs have a fixed action duration Δt. We set this action duration to 10 cycles for realistic action execution in MATISSE. This causes the CAVs to select 6 actions in their action plans for the chosen planning horizon of 60 cycles. The centralized algorithm generates the CAV trajectory instead of CAV actions represented using waypoints. We set the duration between two waypoints to 3 cycles. A total of 20 waypoints are computed for the planning horizon of 60 cycles.

C. Evaluation results

For the evaluation, we perform 60 experiments for each scenario for each of the AirCap, ASM, DeCoC-MCTS and the centralized algorithms and calculate their success rate values. Each experiment's result is considered a success if the algorithm is able to find a valid collision-free solution for the planning horizon, otherwise the result is considered to be unsuccessful.

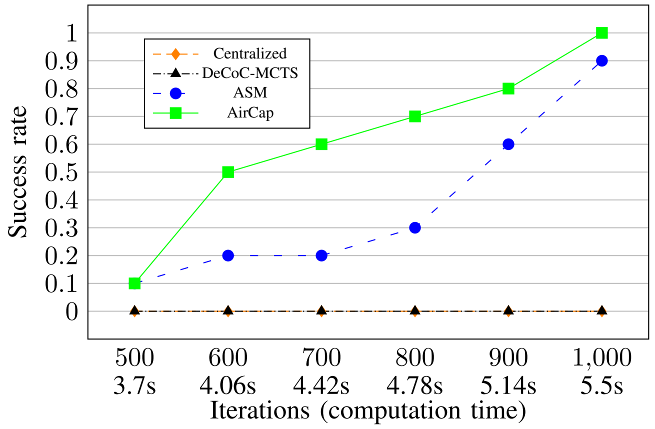


Fig. 5: Scenario 1: The misbehaving vehicle is accelerating from behind a coalition of six CAVs.

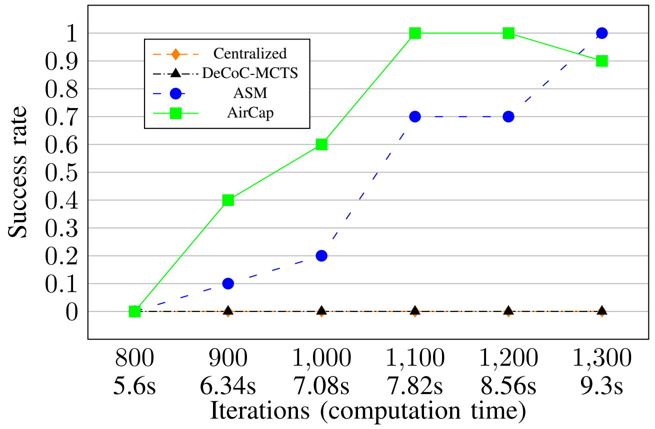


Fig. 6: Scenario 2: The misbehaving vehicle comes to a full stop in front of a coalition of six CAVs.

For AirCap and ASM experiments, we vary the number of MCTS iterations and plot the success rate vs the number of MCTS iterations. One AirCap iteration does not directly correspond to one DeCoC-MCTS iteration, as the computational complexity for both algorithms is different for a single MCTS iteration. We estimate the average computation time of AirCap in seconds for each value of the number of iterations (e.g., 3.7s for 500 iterations of AirCap for scenario 1). Experiments for DeCoC-MCTS and the centralized algorithm are executed for the estimated computation time limits for fair comparison and we plot their success rates vs the computation time limit. Figures 5, 6, and 7 show the success rate vs the number of iterations (or computation time in case of DeCoC-MCTS and the centralized algorithm) for scenarios 1, 2, and 3 respectively.

For scenario 1 and 2 involving six CAVs, the centralized algorithm and DeCoC-MCTS are not able to find solutions within the computation time limits. The reason according to our analysis is that the centralized algorithm and DeCoC-MCTS are not able to find solutions for coalition sizes larger than 4 within computation time limit. For scenario 3 involving a smaller coalition of four CAVs, these algorithms

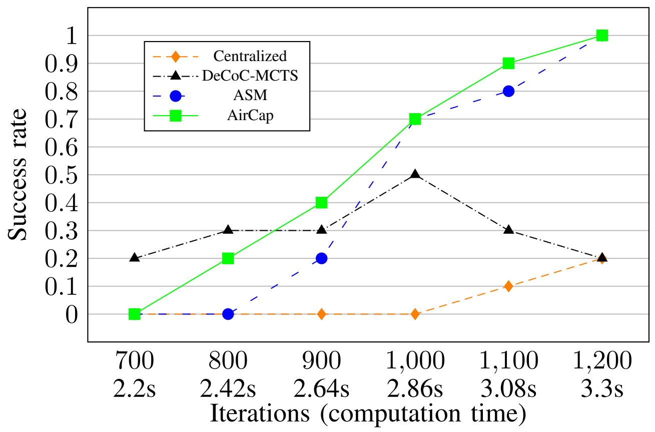


Fig. 7: Scenario 3: The coalition member misbehaves in the middle of a coalition of four CAVs.

are able to find solutions and achieve non-zero success rates.

Scenario 1 and 2 results clearly show the superior performance of AirCap over ASM. For scenario 3, AirCap improves upon ASM with smaller margins. The reason for this behavior is the size of the coalition and CAV positions in the coalition. Scenario 1 and 2 involve larger coalition sizes of 6 than scenario 3. The CAV positions in the coalitions are also more dense in scenario 1 and 2 compared to scenario 3. Large coalition sizes and dense coalition states both limit the number of non-colliding action choices for coalition members, as more CAV action plans lead to collisions among coalition members. In such situations, coalition state based weights updating algorithm significantly improves MCTS algorithm's ability to prioritize CAV action plans. This results in better success rates for AirCap algorithm. Scenario 3 involves smaller number of CAVs and the configuration of CAVs is also less dense. Additionally, the misbehaving vehicle is situated in the middle of the coalition, which limits the action choices for the coalition members, as more CAV action plans lead to collisions with the misbehaving vehicle. In such scenarios, coalition state based reward weights update makes less improvement in prioritizing non-colliding action plans. This leads to relatively smaller improvements in Scenario 3.

Another important finding from experiments is, there is a positive correlation between success rate values and iteration values for AirCap and ASM algorithms indicating better reliability of these approaches, which is not true for DeCoC-MCTS.

VII. Conclusion

In this paper, we presented **A**dapt**I**ve **R**eward for **CA**Vs Action **P**lanning (AirCap), an MCTS-based cooperative action planning algorithm for a coalition of CAVs in emergency scenarios. We proposed a novel idea of dynamically changing CAVs reward function weights for different actions based on resultant coalition states. The proposed AirCap algorithm significantly improves upon its predecessor ASM algorithm. We implemented AirCap in MATISSE and evaluated it against the state-of-the-art centralized and decentralized algorithms. Experimental results show that AirCap's coalition state-based weights update strategy significantly improves the algorithm's success rate than a fixed reward function in large coalitions with dense CAV configurations. AirCap also outperforms competitor centralized and decentralized algorithms.

The presented AirCap algorithm only considers the number of CAVs in the CAV's current neighborhood. We plan to explore using other coalition state attributes to dynamically update the reward function weights.

References

[1] H. Manoochehri and R. Wenkstern, "Dynamic coalition structure generation for autonomous connected vehicles," in *2017 IEEE International Conference on Agents (ICA)*. IEEE, 2017, pp. 21–26.

[2] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[3] C. Andriotis and K. Papakonstantinou, "Managing engineering systems with large state and action spaces through deep reinforcement learning," *Reliability Engineering & System Safety*, vol. 191, p. 106483, 2019.

[4] T. Vodopivec, S. Samothrakis, and B. Ster, "On monte carlo tree search and reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 60, pp. 881–936, 2017.

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[6] D. Lenz, T. Kessler, and A. Knoll, "Tactical cooperative planning for autonomous highway driving using monte-carlo tree search," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016, pp. 447–453.

[7] K. Kurzer, C. Zhou, and J. M. Zöllner, "Decentralized cooperative planning for automated vehicles with hierarchical monte carlo tree search," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 529–536.

[8] K. Kurzer, F. Engelhorn, and J. M. Zöllner, "Decentralized cooperative planning for automated vehicles with continuous monte carlo tree search," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 452–459.

[9] D. Patel and R. Zalila-Wenkstern, "Collaborative collision avoidance for cavs in unpredictable scenarios," in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. IEEE, pp. 1–6.

[10] T. Kessler and A. Knoll, "Cooperative multi-vehicle behavior coordination for autonomous driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1953–1960.

[11] C. Burger and M. Lauer, "Cooperative multiple vehicle trajectory planning using miqp," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 602–607.

[12] M. Düring, K. Franke, R. Balaghiasefi, M. Gonter, M. Belkner, and K. Lemmer, "Adaptive cooperative maneuver planning algorithm for conflict resolution in diverse traffic situations," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2014, pp. 242–249.

[13] S. Manzinger, M. Leibold, and M. Althoff, "Driving strategy selection for cooperative vehicles using maneuver templates," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 647–654.

[14] Z. Wang, Y. Zheng, S. E. Li, K. You, and K. Li, "Parallel optimal control for cooperative automation of large-scale connected vehicles via admm," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1633–1639.

[15] A. Nakamura, Y.-C. Liu, and B. Kim, "Short-term multi-vehicle trajectory planning for collision avoidance," *IEEE Transactions on Vehicular Technology*, 2020.

[16] C. Boutilier, "Sequential optimality and coordination in multiagent systems," in *IJCAI*, vol. 99, 1999, pp. 478–485.

[17] B. Torabi, M. Al-Zinati, and R. Z. Wenkstern, "Matisse 3.0: A large-scale multi-agent simulation system for intelligent transportation systems," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer, 2018, pp. 357–360.