2021 IEEE Intelligent Vehicles Symposium (IV)
July 11-17, 2021. Nagoya, Japan

Scalable Monte Carlo Tree Search for CAVs Action Planning in Colliding Scenarios

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

Abstract— **Connected and autonomous vehiles (CAVs) require an effective cooperative action planning strategy in an emergency situation. Monte Carlo Tree Search (MCTS) is a promising planning technique for such problems with large state spaces. However, traditional MCTS-based techniques do not scale well with the number of vehicles. In this paper, we present a novel MCTS-based cooperative action planning algorithm for CAVs driving in a coalition formation. Our proposed algorithm improves the reliability and the scalability of MCTS. Explicit communication is used to ensure that mitigation action plans chosen by the CAVs are conflict-free when possible. We perform the evaluation of the proposed algorithm in a large scale multi-agent based traffic simulation system. Our simulated experiments show that our approach improves upon current state-of-the-art centralized and decentralized algorithms.**

I. Introduction

Cooperative action planning for connected and autonomous vehicles (CAV) is central to autonomous driving. In emergency situations, when a CAV misbehaves due to technical problems or unexpected environmental conditions, it is essential for neighbor CAVs to find action plans to avoid collisions. A CAV action plan is a sequence of actions such as acceleration or lane change. In this paper, we consider the problem of generating cooperative action plans for a coalition of CAVs in colliding situations. A CAV coalition is a group of CAVs that drive together for information sharing and mutual support. CAV coalitions improve roadway safety, reduce energy consumption and decrease traffic congestion [1].

Cooperative action planning for CAVs is a complex task that involves decision-making for a multi-agent system. Single-agent decision making algorithms can not be applied as the number of possible solutions grow exponentially with the number of agents. In recent years, a variety of cooperative action planning methods for CAV collision avoidance have been proposed [2], which include centralized, decentralized and partially decentralized methods. In centralized methods, a central server uses the state information sent by all CAVs to solve a centralized optimization problem and finds action plans for all CAVs. Centralized methods deterministically find optimal solutions but scale poorly [3].

Most decentralized and partially decentralized methods formulate the CAV action planning problem as a form of a multi-agent Markov Decision Process(MMDP), which can be solved using different reinforcement learning techniques. However, classical reinforcement learning techniques are not efficient for the CAV action planning problem due to a very large state space [4].

A reinforcement learning technique known as Monte Carlo Tree Search (MCTS) [5] has recently shown promising results in problems with very large state spaces such as the game of Go [6]. MCTS was applied to the CAV action planning problem and performed well in simple scenarios with a small number of CAVs [7] [8]. The performance of MCTS is limited by the branching factor in the MCTS tree. When applied to scenarios with a large number of CAVs, MCTS struggles to scale as the branching factor grows exponentially with the number of CAVs. We have proposed Approximate Simultaneous Move (ASM-MCTS) [9], a modified MCTS-based algorithm that removes the dependence of the branching factor on the number of vehicles. However the approximation uses the random monte carlo sampling approach which reduces the reliability of ASM-MCTS.

We present Improved Simultaneous Move (ISM), a partially decentralized MCTS-based algorithm for CAV action planning. In ISM, we significantly improve the reliability and the scalability of ASM-MCTS by introducing novel designs for all four stages of ASM-MCTS. We propose to intelligently select actions for agents at each node of the MCTS tree by reusing the reward values received in the previous iterations. We implement ISM in MATISSE, a large scale multi-agent traffic simulation system and compare it with the state-of-the-art centralized and decentralized algorithms.

In the next section, we review related works. In Section III, we formalize the problem. In Section IV, we present ISM and in Section V, we present the comparative experimental results which show how ISM improves upon the state-of-the-art centralized and decentralized algorithms.

II. Related works

We classify the proposed collaborative collision avoidance methods according to their planning approach: *non AI-based approaches* and *AI-based approaches*. Each of the category can be further categorized as *centralized*, *decentralized*, or *partially decentralized*. In centralized approaches, a centralized server is responsible for deriving collision avoidance decisions for vehicles. In decentralized approaches, each CAV makes the collision avoidance decision in isolation. In partially decentralized approaches, decision-making is hierarchical and is performed at two levels: at the vehicle level and at the coalition leader level.

[1]Dhruvkumar Patel is with Department of Computer Science, University of Texas at Dallas, 800 W Campbell Rd, Richardson, Texas 75080-3021. dhruv@utdallas.edu

[2]Rym Zalila-Wenkstern is with Department of Computer Science, University of Texas at Dallas, 800 W Campbell Rd, Richardson, Texas 75080-3021. rymw@utdallas.edu

2021 IEEE Intelligent Vehicles Symposium (IV) | 978-1-7281-5394-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/IV48863.2021.9576040

A. non AI-based approaches

Centralized: Most conventional optimization-based approaches fall into this category. These approaches formulate the trajectory planning problem for CAVs as a single, global optimization problem. [10] propose to use Mixed Integer Linear Programming (MILP) with the collision constraints. [11] propose to use Mixed Integer Quadratic Programming (MIQP) with the collision constraints. [12] propose a graph theory based optimization approach. Each vehicle defines its reachable target points (RTPs). The server checks these points for collisions and defines safe target points (STPs) for each vehicle, from which final trajectories are generated. [13] propose to use a pre-defined maneuver template based approach. A maneuver template describes continuous dynamics of a set of vehicles. The server checks the off-line calculated maneuver templates to match with the current traffic scene. The matching maneuver template with the lowest cost specifies the cooperative maneuver. [14] propose a cloud computing based solution that uses parallel processing. The trajectory planning problem is formulated as a centralized Quadratic Programming (QP) problem, which is decomposed using Alternating Direction Method of Multipliers (ADMM) [15] and is solved on a network of computing nodes. [16] propose to formulate the trajectory generation problem as an integer programming problem. The optimization objective is to minimize the travel time. The integer programming problem is solved using a SAT solver, which provides the formal guarantee of the 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 at every third or every sixth timestep instead of at every timestep. This approach is evaluated on scenarios involving up to 5 vehicles. The above mentioned centralized approaches generally provide optimal solutions to the formulated optimization problems, however they are computationally expensive and not scalable.

Decentralized: [17] proposes a brute search method to find trajectories for a group of CAVs. Each CAV computes a list of all possible maneuvers along with the associated costs and shares it with other CAVs. Each CAV then performs an exhaustive search to find non-colliding plan for the whole group. This approach is not scalable due to the redundant and expensive computation performed by each CAV.

B. AI-based approaches

Decentralized: In AI-based approaches, each CAV selects an action by learning values of all actions in the current state and picking the action with the highest value. Most AI-based approaches are either decentralized or partially decentralized. In decentralized approaches, cooperation is achieved implicitly using a cooperative reward function. Decentralized approaches rely on the proposed algorithm's ability to find a Nash equilibrium, i.e., no single agent can perform a different action yielding a higher reward. However a Nash equilibrium isn't always guaranteed and this severely impacts the reliability. On contrary, partially decentralized approaches achieve cooperation explicitly using V2V communication. This approach is much more reliable, in that a non-colliding action plan is always chosen when it's possible. [7] proposes a decentralized approach using Monte Carlo Tree Search (MCTS) [18] with implicit cooperation. Possible actions are chosen from a discrete action set. The approach was evaluated on scenarios involving up to three cooperative vehicles. [8] proposes DeCoH-MCTS, a **de**centralized **co**operative **h**ierarchical MCTS-based approach with implicit cooperation. The concept of macro-actions is proposed which consist the sequences of primitive actions. Macro-actions reduce the effective MCTS tree depth. The evaluation is performed against the scenarios involving up to three vehicles. [19] proposes DeCoC-MCTS, a **de**centralized **co**operative **c**ontinuous MCTS-based approach with implicit cooperation. This method proposes to use continuous action-spaces for CAVs to generate flexible trajectories. Authors propose two techniques to deal with the infinite action-space size, namely kernel update and guided search . In kernel update, action similarity is used to learn the values of similar actions together. In guided search, size of the finite action-set for each vehicle is fixed initially and gradually increased as more iterations are performed. The evaluation is performed against the scenarios involving up to three vehicles. In their later work [20], authors propose a preprocessing step for an MCTS based algorithm. A heuristic model over vehicle actions is learned from a synthetic data generated using simulations. The learned model is plugged back into DeCoC-MCTS to steer the algorithm towards more promising areas of the action space. The authors show that MCTS with heurisic model has higher success rate than the baseline MCTS in some scenarios. MCTS-based approaches have the limitation of scalability as the MCTS tree size grows exponentially with the number of vehicles and the size of the joint action-space. To our knowledge, none of the proposed decentralized algorithms are evaluated using a multi-agent based traffic simulation system. As such their evaluation lacks the simulation of real world multi-agent system properties such as autonomy, local views and partial knowledge.

Partially Decentralized: We have proposed ASM-MCTS [9] **a**pproximate **s**imultaneous **m**ove MCTS-based algorithm with explicit cooperation and hierarchical decision-making. ASM-MCTS reduces the MCTS tree size exponentially compared to other MCTS-based algorithms by removing the dependence on the joint action-space size. This increases the scalability of the MCTS algorithm. However it also introduces the approximation in learning the action values, which at times reduces the reliability.

In this paper, we present a scalable and reliable vehicle-level decision-making algorithm for the partially decentralized approach. We propose a hierarchical decision-making approach where the decision-making is performed at the vehicle level and at the coalition leader level. The unique contributions of our approach are the following:

- We present Improved Simultaneous Move (ISM)

TABLE I: CAV Actions

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

MCTS-based algorithm with novel designs for all four stages of MCTS: selection, expansion, simulation and backpropagation.

- We implement ISM in MATISSE, a large scale multi-agent based traffic simulation system.
- We conduct extensive simulated experiments to show that ISM improves upon the state-of-the-art centralized and decentralized algorithms in reliability and scalability metrics.

III. Model Definition

In the remainder of this paper the terms *agent* and *CAV* are used interchangeably. We consider a coalition $\mathcal{C}$ of n CAVs is navigating on a highway. A coalition is formed when CAVs come in close proximity of each other. $\mathcal{C}$'s agents are called *coalition members*. One coalition member is chosen as a *leader* and is responsible for coalition management. Coalition formation and leader assignment algorithms are adapted from our previous work [21].

A CAV is defined by its ID $i \in \mathcal{C}$ and 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. A coalition joint state is denoted as $s = \{s^i\}_{i \in \mathcal{C}}$.

Table I lists CAV's possible actions. 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 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.

Based on current CAV technologies, we assume that each CAV is equipped with perception sensors such as radars (front and back) and 360°Lidar, which provide short and long range 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., member or leader) and includes a CAV's state s^i, the coalition joint state $s = \{s^i\}_{i \in \mathcal{C}}$, a warning or an action plan α_h^i.

We formulate the problem of decentralized cooperative planning for communicating CAVs as a Multi-agent Markov Decision Process (MMDP) [22]. Unlike a conventional MMDP where each agent considers the immediate reward for joint actions, our approach focuses on maximizing the overall coalition reward based on CAVs' action plans for a horizon. Unlike decentralized planning with implicit cooperation using macro-actions [8], our approach uses explicit V2V communication to coordinate between all members of a coalition. MMDP 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'.

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 estimate Q^* values for the next h consecutive actions, and possible start states for those actions. Each CAV i derives all possible mitigation action plans for horizon h, and ranks the plans using the Q^* values. Each CAV i sends the ranked plans to the coalition leader using V2V communication. The coalition leader analyzes the prioritized mitigation action plans, and deliberates to find, an action plan α_h^i for each CAV i such that the collision constraints are satisfied when possible.

IV. Algorithms

A. Monte Carlo Tree Search and its variants

Monte Carlo Tree Search (MCTS) is an algorithm to solve MDPs using a combination of a tree-search algorithm and Monte-carlo simulations. MCTS tree consists of nodes referring to MDP states and edges referring to MDP actions. One iteration of MCTS algorithm consists of four steps: Selection, Expansion, Simulation and Backpropagation [18]. In the selection step, nodes are selected using some heuristic function starting from the root node until a leaf node is found. In the expansion step, the selected leaf node is expanded with all the possible actions available at that state and one of the children is selected. In the simulation step, a simulation is performed from the selected child node until a terminal node is reached. In the backpropagation step, reward value received at the end of the simulation is backpropagated from the leaf node to the root node. MCTS is an anytime algorithm so it can be run either until a fixed number of iterations or until the computational budget is exhausted. When MCTS is applied to a multi-agent scenario (MMDP) in which multiple agents perform their actions simultaneously, it's known as Simultaneous Move MCTS (SM-MCTS) [23]. In SM-MCTS algorithm, MCTS tree size increases exponentially with the number of agents and with the size of each agent's individual action space [9]. Thus it becomes impractical and very hard

to find a solution for problems with large action-agent sets as we show in Section V.

In our previous work [9], we proposed Approximate Simultaneous Move Monte Carlo Tree Search(ASM-MCTS), an MCTS-based algorithm which reduces the MCTS tree size exponentially compared to SM-MCTS. This makes ASM-MCTS more suitable for problems with large action-agent sets than SM-MCTS. ASM-MCTS achieves exponentially smaller tree size by reducing the branching factor of the tree. The branching factor in ASM-MCTS is equal to the size of the individual action space instead of the size of the joint action space in SM-MCTS. MCTS tree nodes refer to the joint states and edges refer to the current agent's individual actions. As each edge only refers to an individual action for the current agent, actions for other agents are determined using random sampling strategy. This methodology introduces approximation in estimation of each node's MCTS statistics and may potentially decrease the reliability of the results. We confirm this outcome using extensive simulated experiments in Section V. In this paper, we present Improved Simultaneous Move (ISM) algorithm that is both reliable and scalable.

B. Weighted reward function

For ISM, we propose a weighted reward function. Individual CAV i's reward $r^i_{t_k}$ at time t_k consists of several weighted reward terms as following:

$$r^i_{t_k} = r^i_{collision} + w \cdot r^i_{comfort} \tag{1}$$

Here $r^i_{collision}$ refers to the penalty inflicted upon simulating a colliding action. $r^i_{comfort}$ includes several reward function terms that are weighted by the weight vector w. These reward terms include speed deviation, acceleration, lane changes, lane deviation from the initial lane, and full stop. Cooperative joint reward $r^i_{t_k,coop}$ is defined in terms of the individual reward terms and is computed by CAV i as follows:

$$r^i_{t_k,coop} = r^i_{t_k} + \sum_{j \in \mathcal{C} \backslash i} \lambda \cdot r^j_{t_k} \tag{2}$$

Here λ is the cooperation factor that controls the cooperation of CAV i for other CAVs.

C. Improved Simultaneous Move (ISM) algorithm

In this paper, we introduce Improved Simultaneous Move (ISM), an MCTS-based algorithm that is both reliable and scalable compared to ASM-MCTS. To this effect, we propose to intelligently select actions for other agents by reusing the reward values received in the past iterations. ISM also inherits its ASM-MCTS algorithm's exponential reduction of MCTS tree size achieved using the reduction of the branching factor. ISM modifies all four ASM-MCTS steps: selection, expansion, simulation and backpropagation. In the remainder of this section, we consider that a coalition member *i* receives a misbehaving vehicle warning and deliberates to find mitigation action plans for the planning horizon *h* taking into account the current states of all coalition members. Construction of the MCTS search tree is the key component of the individual decision making task.

1) ISM search tree structure: In ISM algorithm executed by CAV *i*, a node *n* in MCTS search tree includes following parameters at time t_k (t_k represents the time at which action $a^i_{t_k}$ from the mitigation action plan α^i_h is executed):

- $n.s_{t_k}$: the joint state of the coalition at time t_k
- $n.s^i_{t_k}$: the individual state of CAV i at time t_k
- $n.a^i_{t_{k-1}}$: the individual action for CAV i taken at time t_{k-1}
- $n.V$: a map indexed by $(j, a^j_{t_{k-1}}), \forall j \in \mathcal{C} \backslash i$ that stores the sum of rewards received while simulating action $a^j_{t_{k-1}}$ for CAV j at time t_{k-1} in all simulations that include node n
- $n.C$: a map indexed by $(j, a^j_{t_{k-1}}), \forall j \in \mathcal{C} \backslash i$ that stores the number of simulations in which action $a^j_{t_{k-1}}$ was selected for CAV j at time t_{k-1} in all simulations that include node n
- $n.\nu$: a value that corresponds to the sum of cumulative rewards of all simulations that include node n
- $n.cnt$: the visit count of node n
- $n.n_{parent}$: the parent node of node n
- $n.R_{temp}$: a map indexed by IDs of the coalition members and corresponds to the reward values received at this node by each coalition member during the current MCTS iteration
- $n.n_{parent}$ is the parent node of node n
- $n.n_{children}$ is the set of child nodes of node n

We initialize the root node n_μ of the search tree with the current joint state $n_\mu.s_{t_1}$, CAV i's current individual state $n_\mu.s^i_{t_1}$. Other variables $n_\mu.V$, $n_\mu.C$, $n_\mu.a^i_{t_0}$, $n_\mu.a_{t_0}$, $n_\mu.n_{parent}$, $n_\mu.n_{children}$ are set to null where as $n_\mu.\nu$ and $n_\mu.cnt$ are set to zero. The most important functions of ISM algorithm executed by CAV i are outlined in Algorithm 1.

Algorithm 1 ISM

Input: n_μ, A

1: $k \leftarrow 1$
2: **while** maximum number of iterations are not executed **do**
3: $\quad \langle n_l, k, \mathcal{R} \rangle \leftarrow$ ISM-SelectionPolicy(n_μ, k, A)
4: $\quad$ **if** $n_l.cnt \neq 0$ **then**
5: $\quad\quad \langle n_l, k, \mathcal{R} \rangle \leftarrow$ ISM-ExpansionPolicy$(n_l, k, A, \mathcal{R})$
6: $\quad$ **end if**
7: $\quad \mathcal{R} \leftarrow$ ISM-SimulationPolicy$(n_l, k, \mathcal{R})$
8: $\quad$ ISM-BackpropagationPolicy$(n_{l,k,\mathcal{R}})$
9: **end while**

We describe the four main steps of ISM algorithm in the next sections.

2) ISM selection policy: Starting from the root node n_μ, we consecutively select nodes in each step using UCB1 algorithm [18] applied over $n_{children}$ using node statistics ν and cnt in step 5 and 6. The child node n that is selected automatically determines action $a^i_{t_k}$ for current CAV i. To select actions $a^j_{t_k}$ of other coalition members $j \in \mathcal{C} \backslash i$, we apply UCB1 algorithm over the stored action reward values $V(j)$ and the count values $C(j)$ in step 8. $a^i_{t_k}$ combined with

$a_{t_k}^{\mathcal{C}\backslash i}$ forms the joint action set a_{t_k}, which is then applied to the joint state set s_{t_k} of the parent node n_{parent} to derive n's joint state set $s_{t_{k+1}}$ in step 10. We compute separate reward values for all coalition members and store it in R_{temp} array. The values in R_{temp} are then used to update the cumulative reward $\mathcal{R}$.

Algorithm 2 ISM-Selection Policy

Input: n_μ, k, A
Output: $n, k, \mathcal{R}$
1: $n \leftarrow n_\mu$
2: $\mathcal{R} \leftarrow 0$
3: **repeat**
4: $a_{t_k}^i \leftarrow$ UCTAction$(A^i, n.n_{children}.v, n.n_{children}.cnt)$
5: $n \leftarrow$ node with $a_{t_k}^i$ action from $n.n_{children}$
6: **for** j in $\mathcal{C}\backslash i$ **do**
7: $n.a_{t_k}^j \leftarrow$ UCTAction$(A^j, n.V(j), n.C(j))$
8: **end for**
9: $n.s_{t_{k+1}} \leftarrow$ ComputeState$(n_{parent}.s_{t_k}, n.a_{t_k})$
10: $n.R_{temp} \leftarrow$ ComputeReward$(n_{parent}.s_{t_k}, n.s_{t_{k+1}}, n.a_{t_k})$
11: $\mathcal{R} \leftarrow \mathcal{R} + n.R_{temp}(i)$
12: **for** j in $\mathcal{C}\backslash i$ **do**
13: $\mathcal{R} \leftarrow \mathcal{R} + \lambda \cdot n.R_{temp}(j)$
14: **end for**
15: $k \leftarrow k + 1$
16: **until** n is a leaf node

3) ISM expansion policy: To expand a node n in the tree, we first compute new individual states $s_{t_{k+1}}^i$ using the current individual state $s_{t_k}^i$ and each valid individual CAV action $a_{t_k}^i$. The validity of actions is determined using the preconditions listed in Table I. For an individual action space size of 5, up to five child nodes are added in $n_{children}$. The task of computation of joint states $s_{t_{k+1}}$ for these child nodes as well as the selection task of the child node is left to *ISM-SelectionPolicy* call in step 6, which uses UCB1 algorithm to select the actions for the other coalition members and computes the corresponding joint states.

Algorithm 3 ISM-Expansion Policy

Input: $n, k, A, \mathcal{R}$
Output: $n_l, k, \mathcal{R}$
1: **for** $a_{t_k}^i$ in A^i **do**
2: $s_{t_{k+1}}^i \leftarrow$ computeState$(n.s_{t_k}^i, a_{t_k})$
3: $n_{child} \leftarrow$createNode$(a_{t_k}^i, s_{t_{k+1}}^i)$
4: $n.n_{children} \leftarrow n.n_{children} \cup n_{child}$
5: **end for**
6: $\langle n_l, k, \mathcal{R} \rangle \leftarrow$ ISM-SelectionPolicy(n, k, A)

4) ISM Simulation policy: We perform the simulation step starting from a leaf node n_l of the MCTS tree. We select the joint action set a_{t_k} randomly from the joint action space A. The next state $s_{t_{k+1}}$ is derived from the current node's joint state s_{t_k} and the sampled joint action a_{t_k}. Using the derived joint state, we create a new child node n_{child}. We compute separate reward values for all coalition members in R_{temp} array which is later used to update the cumulative reward $\mathcal{R}$. Here, note that n_{child} is not added to the MCTS tree. This procedure is performed until a terminal node is found. A node is considered terminal when a collision among the coalition members is detected or the planning horizon h is reached.

Algorithm 4 ISM-Simulation Policy

Input: $n, k, \mathcal{R}$
Output: $\mathcal{R}$
1: **repeat**
2: $a_{t_k} \leftarrow$ RandomSelection(A)
3: $s_{t_{k+1}} \leftarrow$ ComputeState$(n.s_{t_k}, a_{t_k})$
4: $n_{child} \leftarrow$ ExpandSingleChildNode(n, a_{t_k})
5: $R_{temp} \leftarrow$ ComputeReward$(n.s_{t_k}, n_{child}.s_{t_{k+1}}, n_{child}.a_{t_k})$
6: $\mathcal{R} \leftarrow \mathcal{R} + R_{temp}(i)$
7: **for** j in $\mathcal{C}\backslash i$ **do**
8: $\mathcal{R} \leftarrow \mathcal{R} + \lambda \cdot R_{temp}(j)$
9: **end for**
10: $n \leftarrow n_{child}$
11: $k \leftarrow k + 1$
12: **until** n is a terminal node

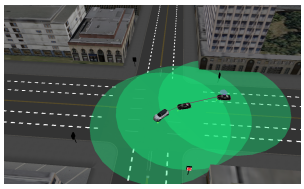
5) ISM Backpropagation policy: During the backpropagation step, the cumulative reward computed at the end of the simulation step is used to update $n.\nu$ and $n.cnt$ values at each node in the reverse path from the leaf to the root. Values in V and C are also updated using the values in $R_t emp$ for the actions that were selected during the current iteration for the other coalition members. Additionally, all the joint state values s as well as the action values of the other coalition members $a_{t_k}^{\mathcal{C}\backslash i}$ for all the nodes along the reverse path are set back to $null$ in their respective state parameter vectors except at the root node.

Algorithm 5 ISM-Backpropagation Policy

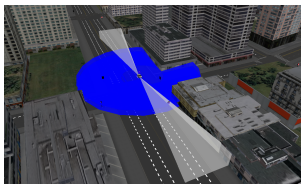
Input: $n, k, \mathcal{R}$
1: **while** $n \neq n_\mu$ **do**
2: $n.cnt \leftarrow n.cnt + 1$
3: $n.\nu \leftarrow n.\nu + \mathcal{R}$
4: **for** j in $\mathcal{C}\backslash i$ **do**
5: $n.V(j, n.a_{t_k}^j) \leftarrow n.V(j, n.a_{t_k}^j) + n.R_{temp}(j)$
6: $n.C(j, n.a_{t_k}^j) \leftarrow n.C(j, n.a_{t_k}^j) + 1$
7: **end for**
8: $n.s_{t_k} \leftarrow$ null
9: $n.a_{t_k}^{\mathcal{C}\backslash i} \leftarrow$ null
10: $n \leftarrow n.n_{parent}$
11: $k \leftarrow k - 1$
12: **end while**

At the end of ISM execution, each CAV i estimates the state-action value function Q^* at node n using following equation:

$$Q^*(n_{parent}.s_{t_{k-1}}^i, n.a_{t_{k-1}}^i) \approx \frac{n.\nu}{n.cnt} \tag{3}$$

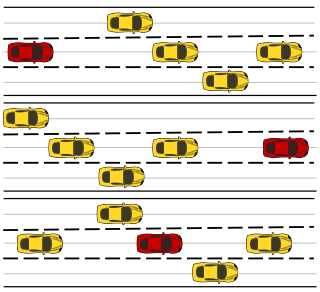


(a) Vehicle agents communicate within communication radius in MATISSE



(b) Vehicle agent with long range RADAR and short range LiDAR in MATISSE

Fig. 1: Realistic simulation of CAVs in MATISSE



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

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

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

Fig. 2: Placement of coaltion members (yellow) and the misbehaving vehicle (red) in tested scenarios

Each path in the MCTS tree from the root node to a leaf node represents one mitigation action plan. 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].

V. Experimental results

In this section, we evaluate the reliability and scalability of the proposed ISM algorithm and compare it with the state-of-the-art centralized algorithm proposed by [16], the leading decentralized algorithm DeCoC-MCTS proposed by [19] and the ASM-MCTS algorithm [9]. We first discuss the multi-agent based simulation framework, the experimental setting used to run the simulated experiments and then present the results.

A. MATISSE

We created a CAV model and tested the algorithms in MATISSE 3.0, a microscopic multi-agent based traffic simulation system developed at the MAVS lab at UT Dallas [24]. The virtual CAV is implemented as an autonomous decision-making agent. It has the capability to dynamically sense its surroundings through simulated sensors (see Fig. 1b) while communicating with other virtual CAVs through simulated V2V communication (see Fig. 1a). During the simulation, the user can modify a CAV's properties (e.g., sensor range) and behavior (e.g., speed) and witness the outcome in simulated real-time.

B. Experimental setting

We have implemented and tested ISM, ASM-MCTS, DeCoC-MCTS, and the centralized algorithm by Nakamura et al. in MATISSE, the agent-based traffic simulator [1]. In our simulated experiments, a group of CAVs are navigating on a three-lane straight highway at the speed of 2 units/cycle. A unit refers to a unit of length in MATISSE, and a cycle refers to one full simulation cycle. During the simulation, we alter the behavior of one CAV (e.g., increase speed or slow down) in a way that cause it to misbehave and have a collision with one of the coalition members after the "Time To Collide" (TTC) cycles. When this misbehaving vehicle is detected through simulated sensors, each coalition member executes the selected algorithm to find its collision-avoiding trajectory for a planning horizon h. The TTC values in these simulated experiments are uniformly selected from $\{10$ cycles, 15 cycles, 30 cycles$\}$ to simulate different emergency levels.

At the maximum TTC of 30, the collision is detected after 30 cycles. We set the planning horizon $h = 60$ cycles, twice the maximum TTC. The reason is that the coalition members should not only avoid the detected collision at TTC=30, but avoid possible collisions even after TTC until the coalition is out of the danger with respect to the misbehaving vehicle.

For ISM, ASM-MCTS, and DeCoC-MCTS, vehicle actions can only have a fixed duration Δt. We set this action duration to 10 cycles for realistic action execution in MATISSE. Since the planning horizon is 60 and the action duration is 10, each CAV deliberates to select next 6 actions. The centralized algorithm by Nakamura et al. generates waypoints at each time step that represent the vehicle trajectory . The duration between two time steps is fixed in this algorithm. We set the time step duration to 3 cycles. For the planning horizon of 60 cycles and time step duration of 3 cycles, a total of 20 waypoints are computed by the centralized algorithm.

C. Reliability results

We define reliability as a collision avoidance algorithm's ability to find a non-colliding action plan for CAVs. Since the emphasis of the simulated experiments in this section is on the reliability as opposed to the scalability, we consider a small coalition of four vehicles. We test the algorithms on three sets of scenarios which differ on the position and speed of the misbehaving vehicle (see Fig. 2), as listed below:

- *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 coalition members and either speeds up at 5 units/cycle or comes to a full stop at 0 units/cycle.

We performed a total of 1056 simulated experiments, 264 experiments for each of the ISM, ASM-MCTS, DeCoC-MCTS and the centralized algorithm by Nakamura et al. Out of 264 experiments for each algorithm, 108 experiments

[1]Demo videos available at `http://www.utdallas.edu/~dhruv/IVdemos`

were performed for each of scenario 1 and scenario 2, and 48 experiments were performed for scenario 3. In experiments for scenarios 1 and 2, we were able to consider various distances between the misbehaving vehicle and the coalition vehicles. For scenario 3, the number of experiments is smaller, because the misbehaving vehicle's position (i.e., inside the coalition) limits the number of options.

The processing times to compute the solution are set to 5s, 10s, 20s, 40s. Although these values are not real-time implementable in emergency situations, they are comparable to the processing times of current state-of-the-art centralized and decentralized algorithms. Each experiment's result is considered a success if the algorithm is able to find a collision-free solution for the planning horizon, otherwise the result is considered to be unsuccessful. Fig. 3 shows the success rate vs the processing time for each algorithm for scenarios 1-3.

For scenario 1, we can see that the centralized algorithm's success rate is close to 1. ISM has a significantly higher success rate than the ASM-MCTS and DeCoC-MCTS. For higher processing time values, ISM even outperforms the centralized algorithm by Nakamura et al.

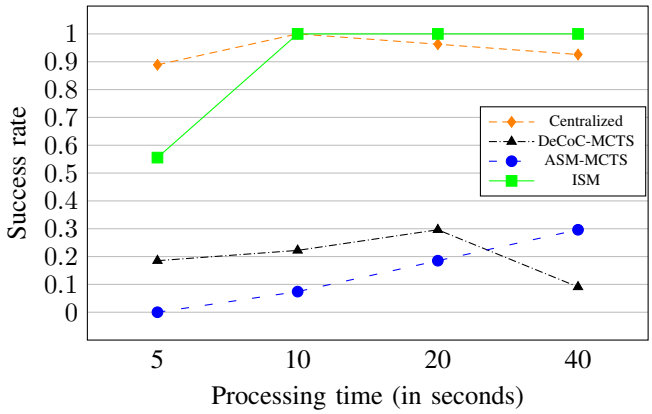
For scenario 2, we can see that the centralized algorithm's success rate is the lowest. In this scenario, lane change movements are essential for the collision avoidance. However the centralized algorithm does not include some important constraints required for checking the collisions that occur during lane change movements, which explains the low success rate. ASM-MCTS has slightly higher success rate than the centralized algorithm. ISM has the highest success rate in this scenario as well.

In scenario 3, the misbehaving vehicle is positioned inside the coalition and either accelerates or comes to a full stop. We can see that the ISM performs the best followed by the centralized algorithm and ASM-MCTS. DeCoC-MCTS gives the lowest success rate for this scenario. An important observation is that ISM has a positive correlation with the allowed processing time, as the success rate always increases with the increase in the processing time.

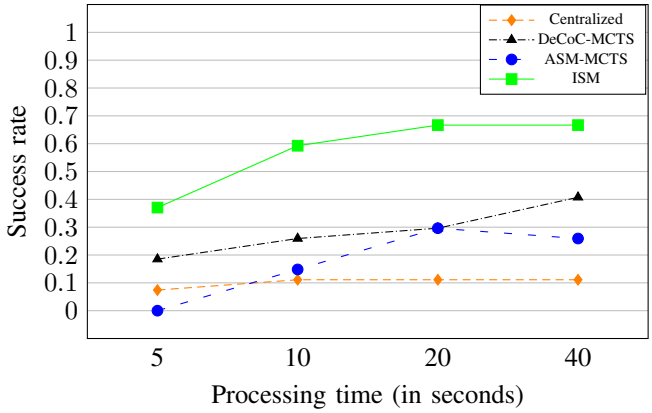
D. Scalability experiments

We define *scalability* as the maximum number of vehicles for which the algorithm can complete its execution in a reasonable amount of time. We aim to compare the scalability of each of the algorithms for different types of misbehavior such as acceleration or deceleration by the misbehaving vehicle. For the simulated experiments performed to compare the scalability of different algorithms, we only consider scenario 1 and 2 as described in Section V-C. We did not consider scenario 3, as it differs from scenario 1 and 2 only in the misbehaving vehicle's position but not in the misbehavior type. We fix the processing time allowed for each algorithm to a maximum of 60 seconds.

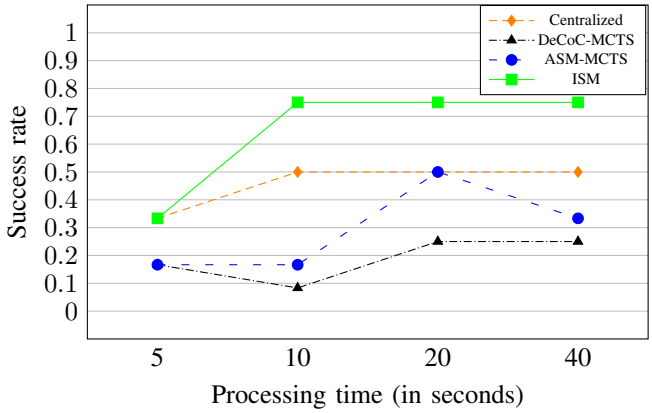
Fig. 4 shows that the centralized algorithm can find solutions for the coalitions of 4 and 3 vehicles for scenarios 1 and 2 respectively. DeCoC-MCTS can find solutions for the coalition of 6 vehicles for both scenarios. These numbers



(a) Scenario 1: The misbehaving vehicle is accelerating from behind a coalition of four CAVs.



(b) Scenario 2: The misbehaving vehicle comes to a full stop in front of a coalition of four CAVs.



(c) Scenario 3: The coalition member misbehaves in the middle of a coalition of five CAVs.

Fig. 3: Reliability results

are hard limits for these algorithms, which means that the computation becomes extremely slow and does not terminate within the allowed processing time. For ASM-MCTS, the plot shows values that are not the hard limits. In other words, finding a solution for a larger number of vehicles that the plotted values is computationally possible, but the found solution may not be collision-free. These values for ASM-MCTS are 7 vehicles for scenario 1 and 6 vehicles

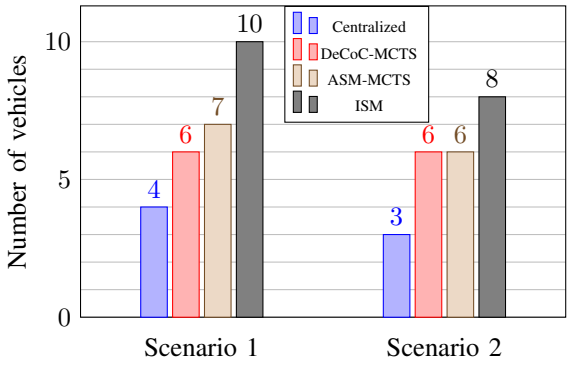


Fig. 4: Scalability results

for scenario 2. ISM considerably increases the soft limit of ASM-MCTS to 10 vehicles for scenario 1 and 8 vehicles for scenario 2. This is a significant improvement over the centralized algorithm by Nakamura et al. and DeCoC-MCTS.

VI. Conclusion

In this paper, we presented a cooperative action planning method for a coalition of CAVs to find non-colliding action plans in the presence of a misbehaving vehicle. In the two-step decision-making method, each CAV first executes ISM to derive ranked mitigation action plans, then the coordinated plans are sent to the coalition leader to ensure that they are collision-free. The proposed ISM algorithm improves the reliability and the scalability of ASM-MCTS by intelligently selecting actions for other agents at each MCTS node. It also adopts the exponential branching factor reduction of ASM-MCTS. We implemented ISM in MATISSE along with the other state-of-the-art centralized and decentralized approaches. Experimental results show that ISM improves upon the centralized and the decentralized approaches in both reliability and scalability metrics.

The presented algorithm only considers a single coalition of CAVs. Our goal is to expand this approach to multiple coalitions of CAVs and compare the performance for multiple coalitions.

References

[1] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2391–2406, 2017.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

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

[17] M. Duering and P. Pascheka, "Cooperative decentralized decision making for conflict resolution among autonomous agents," in *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*. IEEE, 2014, pp. 154–161.

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

[19] 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.

[20] K. Kurzer, M. Fechner, and J. M. Zöllner, "Accelerating cooperative planning for automated vehicles with learned heuristics and monte carlo tree search," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1726–1733.

[21] 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.

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

[23] M. J. Tak, M. Lanctot, and M. H. Winands, "Monte carlo tree search variants for simultaneous move games," in *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 2014, pp. 1–8.

[24] 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.