TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Monte Carlo Tree Search for Autonomous Driving in Lane-Free Traffic Settings

Author:
Pantelis GIANKOULIDIS

Supervisor:
Prof. Georgios CHALKIADAKIS

A thesis submitted in fulfillment of the requirements for the degree of Diploma in Electrical and Computer Engineering

in the

School of Electrical and Computer Engineering

June 19, 2023

"We must not cease from exploration and the end of all our exploring will be to arrive where we began and to know the place for the first time."

T.S.Elliot

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Diploma Thesis

Monte Carlo Tree Search for Autonomous Driving in Lane-Free Traffic Settings

by Pantelis GIANKOULIDIS

Lane-Free traffic is a novel paradigm that lifts the notion of lanes in traffic environments populated (as a first step, only) with autonomous vehicles, resulting in much higher efficiency since the road capacity is better exploited. Despite its novelty, a significant amount of research on lane-free autonomous driving has been already performed. However, well-known Artificial Intelligence (AI) intelligent search and decision planning algorithms have not been yet explored in this setting.

To this end, we expand upon the research in lane-free vehicle movement strategies by introducing a different approach to the problem. Monte Carlo Tree Search (MCTS), a popular search algorithm for decision planning in games, is adopted for the problem at hand. We introduce a formulation for the task of lane-free driving using this algorithm and examine its efficiency under two different settings, which differ with respect to the existence of communication among vehicles, and the very constituents of the basic MCTS algorithm.

In the first setting, each vehicle acts independently from the others according on the formulation of the lane-free environment that is suitable for the MCTS algorithm. The formulation we introduce addresses the two objectives of the vehicles, namely collision avoidance and reaching or preserving a desired speed of choice.

While this approach gives satisfactory results, it does not take into consideration online interactions among vehicles. For every vehicle, other vehicles are observed as moving obstacles with constant speed. If we consider communication among vehicles as well, we can additionally model these interactions and examine their influence in their decision making.

As such, in the second setting we address the multiagent nature of the lane-free environment. For that, we adopt a recently introduced multiagent planning algorithm based on MCTS and Coordination Graphs. Essentially, vehicles exchange messages that affect their planning, consequently resulting in a coordinated decision making process. Then, we provide an extensive set of experiments in order to evaluate our proposed approach under these two settings. Our experimental procedure correspondingly involves two distinct phases. First, the single-agent performance is investigated in scenarios populated with a large number of vehicles in a highway, all employing the MCTS algorithm independently. We observe the performance by evaluating collision occurrences and deviation from the desired speed in demanding scenarios that exceed the capacity of equivalent lane-based environments. Finally, we evaluate the multiagent algorithm in three lane-free scenarios that involve a few vehicles, and showcase its increased coordination capabilities compared to the plain MCTS algorithm at the expense of computational time.

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Περίληψη

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Διπλωματική Εργασία

Δενδρική Αναζήτηση Μόντε Κάρλο για Αυτόνομη Οδήγηση Χωρίς τη Χρήση Λωρίδων Κυκλοφορίας

Του Παντελή ΓΙΑΝΚΟΥΛΙΔΗ

Το περιβάλλον οδήγησης χωρίς λωρίδες κυκλοφορίας είναι ένα πρόσφατα προτεινόμενο παράδειγμα περιβάλλοντος κίνησης οχημάτων, το οποίο αφαιρεί την έννοια των λωρίδων κυκλοφορίας σε περιπτώσεις δρόμων όπου υπάρχουν μόνο αυτόνομα οχήματα και οδηγεί σε μεγαλύτερη εκμετάλλευση της χωρητικότητας του δρόμου. Αν και έχει εισαχθεί πρόσφατα, σημαντική έρευνα έχει εκπονηθεί πάνω στο περιβάλλον οδήγησης χωρίς λωρίδες κυκλοφορίας. Ωστόσο, αλγόριθμοι σχεδίασης βασισμένοι σε βιβλιογραφία τεχνητής νοημοσύνης δεν έχουν ακόμα ερευνηθεί. Σε αυτή την εργασία, επεκτείνουμε την έρευνα όσο αφορά τις στρατηγικές κίνησης οχημάτων σε περιβάλλοντα οδήγησης χωρίς λωρίδες κυκλοφορίας, εισάγοντας μια νέα προσέγγιση στο πρόβλημα. Ο Monte Carlo Tree Search (MCTS), ένας δημοφιλής αλγόριθμος απόφασης σε αλυσίδες Μαρκόβ, εφαρμόζεται στο πρόβλημα.

Εισάγουμε μία νέα πρόταση για το πρόβλημα της αυτόνομης οδήγησης χωρίς λωρίδες κυκλοφορίας με τη χρήση αλγορίθμων σχεδίασης και ερευνούμε την αποτελεσματικότητά του σε δύο διαφορετικές εκδοχές ανάλογα με την δυνατότητα επικοινωνίας μεταξύ των οχημάτων και των ίδιων των επιλεγόμενων αλγόριθμων. Στην πρώτη εκδοχή, κάθε όχημα δρα ανεξάρτητα από τα υπόλοιπα σύμφωνα με μια προσαρμογή του περιβάλλοντος χωρίς λωρίδες κυκλοφορίας κατάλληλη για τον MCTS αλγόριθμο. Η πρόταση που εισάγουμε λαμβάνει υπόψη τους δύο αντικειμενικούς στόχους των οχημάτων εν κινήσει, δηλαδή την αποφυγή συγκρούσεων και την κίνηση με μια προεπιλεγμένη επιθυμητή ταχύτητα. Μολονότι η προσέγγιση αυτή δίνει ικανοποιητικά αποτελέσματα, δεν λαμβάνει υπόψη της την αλληλεπίδραση μεταξύ των οχημάτων. Για κάθε όχημα, τα υπόλοιπα οχήματα είναι απλώς εμπόδια κινούμενα με σταθερή ταχύτητα. Εισάγωντας την επικοινωνία μεταξύ οχημάτων, μπορούμε επιπλέον να υπολογίσουμε αυτές τις αλληλεπιδράσεις και την επίπτωση τους στη λήψη αποφάσεων.

Λαμβάνοντας υπόψη τα προηγούμενα, εισάγουμε στην δεύτερη εκδοχή, εναν πρόσφατα προτεινόμενο αλγόριθμο για σχεδίαση σε πολυπρακτορικά περιβάλλοντα βασιζόμενο στον MCTS και σε Coordination Graphs. Με βάση τον αλγόριθμο αυτό, τα οχήματα ανταλλάσουν μηνύματα που επηρεάζουν τις αποφάσεις τους, συμβάλλοντας σε μια συντονισμένη διαδικασία αποφάσεων.

Στη συνέχεια, μια σειρά αναλυτικών πειραμάτων μετράει αντικειμενικά την ποιότητα των προτεινόμενων δύο λύσεων. Η πειραματική διαδικασία έχει δύο διαφορετικές φάσεις. Στην πρώτη φάση, η απόδοση του απλού MCTS εξετάζεται σε μια σειρά από σενάρια δρόμων με πολλά οχήματα σε αυτοκινητόδρομο, όπου όλα τα οχήματα εφαρμόζουν τον MCTS ανεξάρτητα. Υπολογίζουμε την αποτελεσματικότητά του μετρώντας τις συγκρούσεις μεταξύ οχημάτων και την μέση απόκλιση από την επιθυμητή ταχύτητα, σε σενάρια που η χωρητικότητα του δρόμου σε οχήματα είναι μεγαλύτερη από αυτή των δρόμων με λωρίδες κυκλοφορίας.

Τέλος, στην δεύτερη φάση, η απόδοση του πολυπρακτορικού αλγορίθμου εξετάζεται για τρεις συγκεκριμένες περιπτώσεις σεναρίων σε σύντομους δρόμους χωρίς λωρίδες κυκλοφορίας με λίγα οχήματα, και δείχνουμε τις ικανότητες συντονισμού σε σύγκριση με τον απλό MCTS αλγόριθμο, με κόστος έναν αυξημένο χρόνο υπολογισμού.

Acknowledgements

I would like to thank from the bottom of my heart my parents for supporting me in this academic journey, with patience and warmly. I would also like to thank my supervisor Prof. Georgios Chalkiadakis, for his insightful supervision during this thesis work. I am grateful for the thesis committee members Prof. Ioannis Papamichail and Prof. Vasileios Samoladas for spending their valuable time for my presentation. Finally, this work wouldn't have been done without the valuable guidance and assistance of Dimitrios Troullinos, whom I would like to specially thank from the bottom of my heart.

Contents

List of Figures

List of Tables

Dedicated to my parents

Chapter 1

Introduction

Academic and industrial research about self driving vehicles is facing a significant increase in popularity during the last years for many valid reasons. According to studies, human error is responsible for about 94% of road accidents [32]. Furthermore, elderly people and people with disabilities are going to benefit from self driving cars since they would be able to have transportation without assistance. This is essential, considering that in the United states only about 57 millions adult people have some disability [13] that prevents them from driving adequately safe.

This need for safe and efficient driving has met during the last couple of years great developments in sensors, actuators, powerful processors and advanced machine learning algorithms, that has led to cars with some automation capabilities. Existing commercial automated cars have already reached the level 3 of driving automation [24], in the six level pyramid shown in Figure 1.1. In order to further proceed towards full driving automation (level 5 of Figure 1.1), academic research has suggested many solutions to challenging problems self-driving vehicles are facing, in all the well known respective fields that such a system integrates; path-planning, motion-controlling, environmental perception and decision making [38], in order to achieve higher levels of automation.

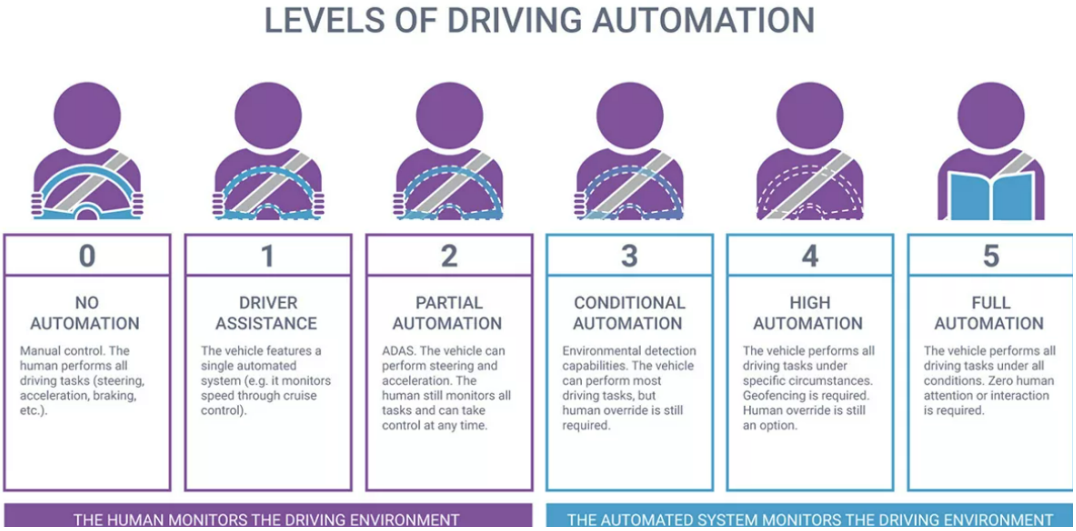


FIGURE 1.1: The levels of driving automation

1.1 Motivation

1.1.1 Sequential decision making for autonomous driving

One particular challenging area of research for the objective of accomplishing safe and efficient self-driving vehicles, is the short-term decision making these systems have to make. A self-driving vehicle has to take decisions sequentially, as the environment in which it interacts changes rapidly, and a decision taken even with a small delay can turn to be catastrophic in extreme circumstances.

Both industry and academia have proposed several methods for the effective sequential decision making in a vehicle traffic environment. In general, there are two main groups that these methods can be classified into, namely classical methods and learning methods [29]. Classical methods involve rule-based and optimization approaches for motion control. Learning methods have gained greater attention during the last years, mainly because of the emergence of new powerful computational technologies and their robustness, in performance terms, in diverse and challenging traffic environments. Huge and extensively labeled real world datasets, together with integrated perception and planning approaches that deep neural networks can perform, have led to effective methods for autonomous driving in conventional lane-based vehicle traffic environments [41].

Other popular approaches for decision making in autonomous vehicles are based in probabilistic models in order to find the most likely traffic scenario, by modelling other vehicle's intentions and take a decision based on that scenario [14]. This approach can be described as a probabilistic approach [41]. In this approach, the nature of real world lane based traffic environments is taken into consideration. In a real world lane based vehicle traffic environment, the agents do not make decisions in the same manner, but many variables can affect their decision; the objective of their driving, their age, the vehicle's state etc. The probabilistic approach incorporates other vehicle's efforts (velocity and acceleration) by forward simulating possible trajectories of these vehicles. The problem is often defined as a partially observable Markov decision process (POMDP) [21], as we will see in detail in Chapter 2. This is the approach that has been followed in this work.

Finally, there is the optimization approach [41] for the decision making in autonomous vehicles. These methods rely on a form of reward or utility function in order to determine the quality of a decision [4].

1.1.2 The Lane-free vehicular traffic environment

A common conception of almost all the aforementioned sequential decision algorithms, is that they assume that the traffic environment is the typical multi-lane environment that exists in modern roads. However, in an environment where all the vehicles would be connected and automated, this assumption may not be true, or even desirable.

The recently proposed TrafficFluid [35] project introduces a novel paradigm for vehicular traffic in the era of connected and automated vehicles (CAVs). The project introduces lane-free traffic environments where lanes are no longer considered, and vehicles can occupy any lateral position. Additionally, vehicle nudging enables vehicles to accommodate overtaking maneuvers of faster vehicles on the back. This traffic paradigm offers several advantages compared to the existing ordinary traffic environments. First, lane-free traffic results in an increase of the available road

capacity, since vehicles are not restricted to lanes and can better utilize the lateral capacity of the road. Second, the anisotropy restriction [43] that is commonly observed in conventional traffic is addressed due to nudging. Typically, in lane-based traffic, vehicle behavior is influenced by traffic downstream (vehicles located in front of the driver). A common observation in congested roads is that congestion "moves" upstream like a wave, due to this forward looking observation. Lane-Free traffic with nudging avoids this undesirable phenomenon, since vehicles on the front can be influenced by the ones on the back. Finally, the lane-free concept allows more freedom in the designing process of a vehicular environment, since these environments do not follow the standard notions of lane-based structures, i.e., car-following and lane-changing maneuvers. The aforementioned property can lead to significant increase in safety and comfort.

1.1.3 The multiagent approach in vehicular traffic environments

The lane-free traffic setting environment has several properties that allow us to consider it as a multi-agent collaborative environment, therefore suitable for this type of algorithms to be applied to the domain.

First, the environment can be considered as a multi-agent environment, since there are typically many vehicles in the road at any given time having similar objectives, properties and abilities [1]. Furthermore, we assert that the vehicles need to collaborate in order to avoid collisions and achieve their desired moving experience in a more efficient manner.

The aforementioned two objectives of collision avoidance and travelling according to a predefined desired speed, can be quantified in local utility functions that each vehicle seeks to maximize. Although the objective of avoiding collisions is common between two vehicles (collision for one of them obviously means collision with the other), the objective of travelling with a desired velocity is potentially adversarial between vehicles in a single-agent approach, because a vehicle travelling with a low desired speed can force another vehicle to travel slower than desired. In other words, one vehicle's increasing of the utility function may cause a huge decreasing in other vehicle's utility function and, as a consequence, the overall system's performance. This means that the objective function of a vehicle must take into consideration other vehicles' desires in order to be more effective.

Considering the aforementioned reasoning, it can be concluded that an assumption that the problem can be formulated as a multi-agent collaborative system is reasonable for the domain in particular.

1.2 Contributions

In this thesis, we introduce two novel formulations based on planning for autonomous driving in lane-free traffic environments. In more detail:

- The problem of planning for autonomous driving in lane-free traffic settings is formulated, considering both single agent settings and multi-agent collaboration.
- The single-agent formulation is applied using the Monte Carlo Tree Search (MCTS) [26] algorithm, an algorithm that creates a decision tree based on bunches of random simulations of the domain, and takes action based on that tree.

In this simple, single-agent planning , each agent acts independently and takes action separately in demanding traffic scenarios that exceed the capacity of lane-based traffic scenarios.

- The effectiveness of a Multi-Agent Variant of the Monte Carlo Tree Search Algorithm is examined in specific collaboration scenarios, and it is compared with the single-agent standard MCTS method of the first formulation.
  In this multi-agent planning, the FV-MCTS algorithm [10] is adopted for the problem, a multiagent MCTS algorithm where all the agents make planning together using global statistics. The decision tree in this case, consists of global states and joint actions.
  In order to make simulations and create the decision tree, FV-MCTS represents local interactions between agents in a coordination graph [12] and takes actions using the iterative MaxPlus message passing algorithm [33] to solve this graph and get the joint action, for every global state.

In Chapter 2 we introduce the theoretical background for the two planning methods we introduce to the domain, along with related work. In Chapter 3 we thoroughly present our proposed solutions for the autonomous driving in lane-free traffic settings problem, for both the single-agent setting and the multi-agent collaboration setting. In Chapter 4 the quality of the aforementioned formulations is examined by extensive experimental evaluation. Finally, in Chapter 5 a brief conclusion with some ideas for future direction of this work are presented.

Chapter 2

Background and related work

Monte Carlo tree search is a very popular algorithm that has been used in a variety of application but, to the best of our knowledge, it has not been used in this domain yet. In Section 2.1.1 we provide the theoretical background of the Monte Carlo Tree Search algorithm. Then, in Section 2.1.2 we introduce an algorithm that tackles multiagent MDPs using planning with Monte Carlo Tree Search in collaborative environments; a modelling that is suitable to a lane-free traffic environment and we later use in our methodology.

Related work in motion planning methods for autonomous driving is extensively discussed in Section 2.2.1, with focus in Monte Carlo Tree Search approaches. Finally, although the experimental and literature background for lane-free traffic is limited due to the novelty of the environment, there are solutions proposed that have made us comprehend the nature and the challenges of the domain better. A brief summary of them is resented in Section 2.2.2.

2.1 Background

2.1.1 Monte Carlo tree search

Monte Carlo Tree Search (MCTS) [7] family of algorithms are among the most popular planning algorithms to date, because they combine the real time planning of a tree search, with the statistical interpretation of the Monte Carlo algorithms. The main idea behind MCTS is to create a decision tree, in order to apply a tree search algorithm to it, not by applying the knowledge for the domain in hand as in the brute force deterministic methods, but by taking random samples from the environment.In other words,they adapt a probabilistic approach in the decision tree problems.

Although it is a new family of algorithms, MCTS have had a lot of success during the last few years, especially in adversarial games. One important milestone about the successful application of the algorithms in this domain, was in 2015, where they were used as part of a Reinforcement Learning approach that beat the world champion in the popular board game Go[40]. They have also been successfully used in a variety of other domains than adversarial games, with notable success.[31].A thorough review of Monte Carlo method and its extensions can be found in [7]

Among the main reasons for this success is that the MCTS algorithms can efficiently explore problems with so many parameters that their search space becomes tremendously huge, problems where traditional deterministic algorithm become extremely slow and therefore useless. The drawback of using these algorithms is that they are probabilistic, therefore they do not always lead to the optimal solution. However, in many cases, they provide solutions that are very close to the optimal, so the fact that they can perform effectively in very large search spaces makes them

compelling for a variety of applications, including our lane-free traffic application, despite the fact that they do not necessarily reach the optimal solution. Other reasons for the recent success of MCTS can be summarized below:

- Aheuristic: It does not require domain knowledge
- Anytime: It can be halted anytime to return the current best estimate
- Assymetric: The algorithm visits the more interesting nodes more often

The aforementioned properties make the algorithms also compelling for using with as part of more advanced learning algorithms.

Markov Decision Process

MCTS [7] can be used for policy-optimization in a finite-size and finite-horizon Markov Decision Process (MDP) [3]. What the algorithm actually does is that it 'captures' the MDP of the domain by forward random sampling from the environment **[2014markov]**.

MDP is a scheme that characterises a fully-observable environment as a stochastic process that holds the Markov property. The Markov property of a stochastic process means that the process is memoryless, which denotes that the next state of the process at a time $t+1$ depends only at the current state at time t, regardless of what states of the process occurred at all the previous time frames $t-1, t-2$.. etc. In other words, the process is fully characterized by the current state only.

In order to construct an MDP, there are four components that must be defined:

- A state space (set) X: It consists of all the possible states of the process.In order to apply the MCTS in this process, this set must be finite.
- An action space (set) A: It consists of all the possible actions that an agent can take.It must be finite in order to apply MCTS in the process.
- A transition probability function P: The probability that an agent's action a,given at a state s_1 results at the state s_2. P: $X \times A \times X$
- A reward function R:The reward of an action a at a state s. R: $X \times A$
- A discount factor $\gamma \in [0, 1)$: The discount term of future rewards, so the expected reward at t is : $R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...$

The state space of a finite-horizon MDP consists of terminal and non-terminal states. A terminal state is a state that the transition probability function returns zero for any other state, given any action and that state.

The goal of a policy-optimization algorithm in an MDP, like the MCTS algorithm, is to determine the policy π that maximizes the (discounted) long-term reward of an agent. A policy is a probability distribution over actions given states $\pi(a_i|s_i) = P(A_t = a_i|S_t = s_i)$. In plain words, it is the probability that an agent takes the action a_i when it founds itself in the state s_i. A policy determines the agent's behaviour at every possible state,therefore it describes the agent's decisions fully.

Finally, in order to decide the best policy, it is essential to formalize what a 'good' state or action is. Given this information, then it is obvious that a good policy would assign higher probabilities to the action A_i that result in higher rewards. We typically estimate the expected long-term (reward) returns using:

- The state value function $V_\pi(s_i)$: It is the expected long-term return of the state s given a policy π
- The action value function $q_\pi(s_i, a_i)$: It is the expected long-term return of an action a_i when taken at a state s_i and following the policy π at the next states.

The action value function and the state value function can be decomposed in two factors: The immediate reward and the discounted future reward when following the policy, weighted by the discount factor γ:

$$V_\pi(s) = E[R_{t+1} + \gamma V_\pi(S_{t+1})|S = s_t] \tag{2.1}$$

$$q_\pi(s, a) = E[R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})|S = s_t, A = a_t] \tag{2.2}$$

The quality of a policy optimization algorithm is related on how well it maximizes the value of the above value functions, so the agent takes actions that are close to the ones that it would take by following the optimal policy. In the case of the MCTS, the policy optimization stems directly from the random simulations of the environment. These simulations approximate the state value and action value of the states and actions by forward random sampling the environment.

The policy that is been followed to solve the MDP after the spaces has been created, is called 'selection policy'. There are two main methods for computing these policies, offline and online methods [36]. Offline method have a predefined policy that is being queried during execution. These policies have computed the best action for every reachable state, therefore the action taken is defined beforehand. Online methods do not separate execution from planing as offline methods do. They compute the next action based on the state they are at, in a certain time, and consider only the states that are reachable from this state.The most popular methods for online planing is MCTS , which is described below in some detail.

The MCTS algorithm

MCTS [7] encodes the MDP it solves as a search tree with nodes and edges. Each node represents a state of the state space of the MDP and each edge represents an action from the action space of the same MDP. A node can be connected to as many edges as the number of state-action pairs involving this state, where the transition probability function of the MDP is non-zero. Alongside with a representation of an MDP state, a node keeps a score associating the utility of the state for the solution of the problem, an approximation of the state-value function of the MDP. In a two-player game for example, this score is usually the long-term possibility of winning if the MDP is at that state.

The novelty of the algorithm is that the state value and action value functions are inferred by the Monte Carlo simulations performed within the subtree of the node, starting from either the root or any children of this subtree. This means that the nodes with greater values are the nodes where the simulations starting from them or their children have got better results. A result of a simulation is regularly the weighted utility of the current and the future states, as it is described in Equation 2.1, when following a simulation policy.

The basic algorithm constructs a tree by taking a continuously iterative process. At each iteration there are four steps as illustrated in Figure 2.1. The iteration process finish after a predefined period of time, or after a certain number of iterations.

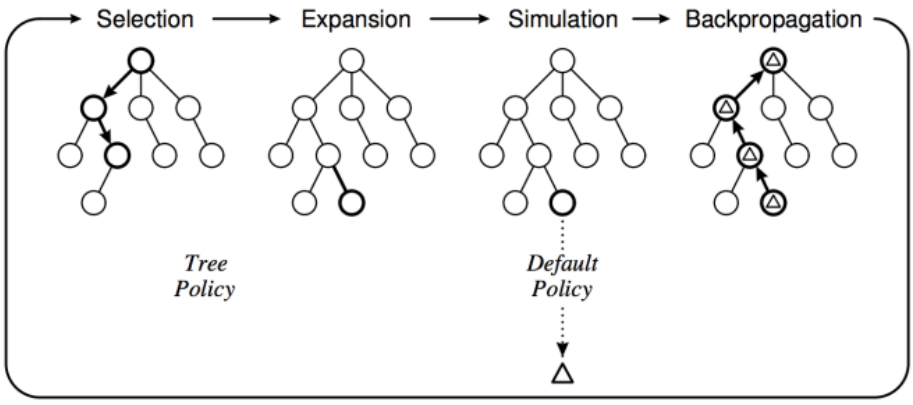


FIGURE 2.1: The phases of one MCTS iteration [7]

Selection In this phase, a leaf node is selected based on a selection policy. There are several proposed policies for selecting a leaf node in the tree. The selection policy is probably the most important thing to consider when designing an MCTS algorithm, because it must maintains the exploration-exploitation trade-off.

Intuitively, the node with the highest estimated value must be selected. A greedy selection would be the optimal policy for solving the MDP. However,the tree constructed so far may not be a precise representation of the MDP. There may exist states with higher value that have not be explored yet. In order to investigate this claim, the algorithm usually adds to the estimated value of the node an exploitation term that is higher for the nodes that has not been visited many times.
The most popular,widely used simulation policies, use the Upper Confidence Bounds applied for tree search, introduced by by Kocsis and Szepesvári [26]. At every node the estimated value of an action is computed as:

$$S_i(a) = Q^{\star}(s,a) + C\sqrt{\frac{ln(N(s))}{N(s,a)}} \tag{2.3}$$

where $N(s)$ is the number of previous visits at the node s, $N(s,a)$ is the number of times the action a was selected when the policy has visited that node and $Q^{\star}(s,a)$ is the current estimation of the action value function $Q(s,a)$ for this state and action that equals to the average returned reward from the simulations. C is a constant that determines the exploration-exploitation trade-off.A small C means that the simulation policy would be more greedy and usually select the nodes with the highest estimated action value function $Q(s,a)$, whereas a big C results in a selection policy that more often explores unvisited states.

Finally, the policy selects at each node the action with the highest $S_i(a)$, except for the final iteration,where it selects the action that the agent would actually perform. In this case, exploitation is pointless and the policy selects greedily the action with the highest estimated value $Q(s,a)$.

Expansion After selecting a leaf node,the algorithm determines whether to expand this node and creates its children or not. This is almost always directly related on how many times the node has been selected so far.If it has not been selected many times in previous iterations, it is better to collect more information about the node.A

common choice is to expand a node if the number of selections of this node is bigger than N,where N is a small natural number.Nodes representing terminal states cannot be expanded.

Simulation After the selection of the node, there is a simulation policy that determines the probability distribution that MCTS samples to generate random actions.The utility of the states visited following the simulation policy, comes heuristically and it depends on the problem in hand.

Backpropagation After receiving a simulation score, the algorithm updates this information to all ancestors nodes of the tree.This way, all the intermediate states from the current state until the state represented in the leaf node update their estimated value functions.

2.1.2 Multi-agent Planning

Multiagent MDPs

A multiagent system can be described by one MDP as described above, with state space S consisting of states S_i describing the entire system, and an action space $A = \prod_{i=1}^{n} A_i$, consisting of all possible combinations of the available actions of the agents [6].

Solving this MDP with the method outlined above is intractable, since the action space grows exponentially with the number of agents. Another approach would be to solve an MDP for every agent independently, like we proposing in the first part of this work in Section 3.1, by using the Monte Carlo tree search methodology independently for every agent. However, these methods often can often lead to suboptimal solutions when considering the possibility of communication among agents. Between the two extremes, many methods have been proposed trying to tackle the trade off between efficiency and optimality using online planning, like factored value MCTS [2] and FV-MCTS with maxplus [10]. In this work, the second method is adopted because it takes better advantage of the anytime nature of MCTS planning.

Other MMDP methods include function decomposition methods, like QMIX [37] and Value decomposition networks [42], while others use parameter sharing [44]. These methods handle the aforementioned trade off between efficiency and optimality through some kind of communication between agents. However, these algorithms don't use MCTS and they are not suitable for a multiagent extension of our work, which depends on MCTS.

Coordination graphs

The concept of coordination graphs was introduced in order to reason about joint values between agents in a Multiagent MDP (MMDP) [12]. It has been empirically show to be an efficient representation for a variety of domains modelled as large MMDPs, because it provides a framework that takes into consideration only local interactions between agents. In a multi-agent setting, each agent is represented as a node in the graph,while an edge between nodes exists only if the individual rewards of the agents depend on each other. The edge in this case,represents the influence an agent has to the other.If the pair of agents do not depend their rewards on each other,there is no edge between them.

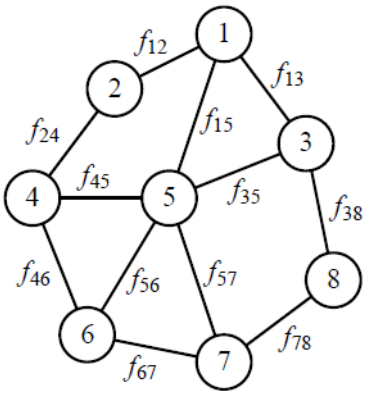


FIGURE 2.2: Example of coordination graph.A node represents an agent and an edge represents an interaction between agents

The global reward of the system is a factored representation of the local rewards of the agents, as shown in Equation 2.4.

$$Q(\overline{A}) = \sum_{i=1}^{n} Q_c(A_i) \tag{2.4}$$

where $\overline{A}$ is the joint action of all the agents and A_i is the action of a component i.

Having this representation, there are several algorithms that have been introduced which attempt to approximate the local component payoffs $Q_c(\overline{A})$ and compute the best joint action $\overline{A}$.

Collaborative MMDP methods with coordination graphs

Obtaining the best joint action in an MMDP with coordination graph is equivalent to computing the maximum posterior solution of an undirected probabilistic graphical model [33]. Unsurprisingly,one of the first methods introduced for solving MMDPs with coordination graphs, is the Variable Elimination (Var-el) algorithm [8], which was initially introduced to solve probabilistic graphical models.

In the context of non-probabilistic methods for solving MMDPs, online setting methods are of particular interest for this work. Kok and Vlassis [33] investigate Var-el for joint action selection in coordination graphs, and introduce the max-plus algorithm as an alternative that is much more scaleable. Amato and Oliehoek propose such a method that combines the idea of MCTS planning with factored values in coordination graphs [2].

The Max-plus Algorithm on Coordination Graphs

An alternative to Var-el algorithm is the Max-plus algorithm, which was initially proposed in the context of taking joint actions in MMDP in coordination graphs in [33]. Max-plus is a popular algorithm for extracting the Maximum Posterior (MAP) configuration of an undirected graphical model. It operates by continuously message passing between nodes, over their corresponding edge, until convergence. Each message is the optimal solution for the two nodes at this time. After the convergence of the messages, each node finds its MAP configuration based on the messages it has received. This operation of finding the MAP configuration is equivalent to finding the best joint action in a coordination graph.

In the context of coordination graphs, max-plus is considered as a local payoffs propagation scheme.The factored global reward 2.4 is a function representation that allows us to do so.

Suppose that there is a coordination graph $G = (V, E)$, with $|V|$ nodes and $|E|$ edges.In every iteration, the nodes perform a belief propagation by sending messages to each other.The value of these messages is computed as follows.

$$\mu_{ij}(a_i) = \max_{a_i} \{f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i)\} \tag{2.5}$$

where $f_i(a_i)$ and $f_{ij}(i, j)$ are the local payoff functions for the agent i and the couple of agents (i,j) respectively. The $\Gamma(i)$ represents all the edges that node is attached to, signifying the locality of interactions in the coordination graphs making them suitable for large multi-agent problems. The message m_{ij} can be regarded as the local payoff function of the node j as it is seen by node i.

After convergence or after a predefined number of iterations, since convergence is not guaranteed by the algorithm, the nodes have computed the messages and can take their optimal action.Each agent computes its optimal action as

$$a_i^* = arg \max_{a_i} \{f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ij}(a_i)\} \tag{2.6}$$

The optimal action is taken based on the local payoff function $f_i(a_i)$ and the sum of the considered by agent i rewards of the agents within its neighborhood.

The joint action is, eventually, the combination of the local optimal actions.

2.1.3 Factored value MCTS with Max-Plus

In the context of multi-agent planning with coordination graphs, Ghoudhurry, Gupta et al. [10] proposed the Factored Value Monte Carlo Tree Search algorithm (FV-MCTS), a solution that uses an extension of the Monte Carlo Tree Search algorithm, suitable for factored representation in coordination graphs, at the top of the max-plus algorithm.

In this multiagent MCTS variant, the state space S consists of the possible global joint states of the system, a combination of all individual states of all agents. The joint action space is the combination of all the possible actions of the agents.In plain words, there is a typical MMDP as described in Section 2.1.2.

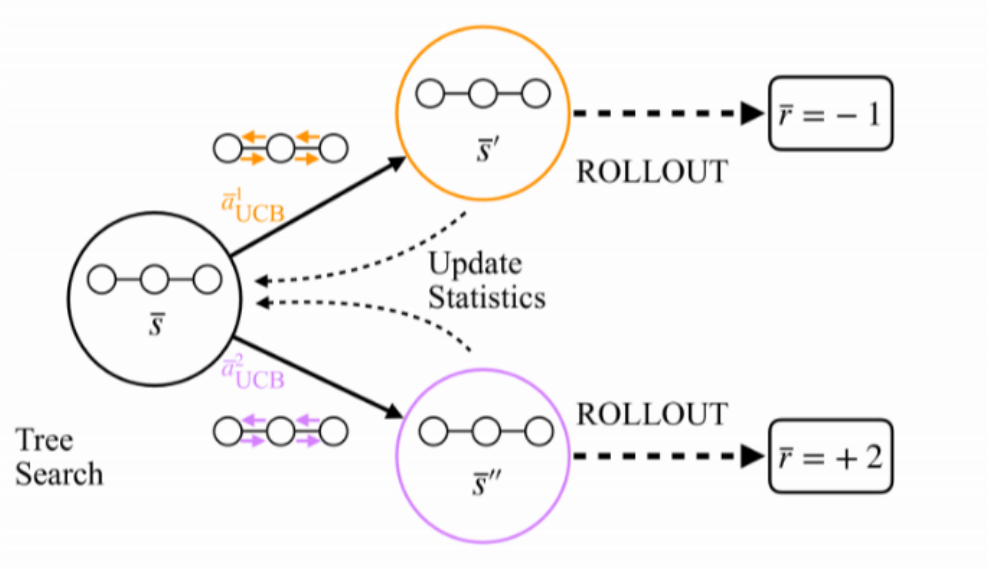


FIGURE 2.3: The algorithm computes the best action $\bar{a}$ for the current joint state $\bar{s}$ by applying the maxplus algorithm in the coordination graph corresponding to the joint state $\bar{s}$

At every state, the global payoff is factored according to the coordination graphs scheme and it is

$$Q(\overline{A}) = \sum_{i \in V} Q_i(a_i) + \sum_{(i,j) \in \mathcal{E}} Q_{ij}(\alpha_i, \alpha_j) \tag{2.7}$$

Here, $Qij(a_i, a_j)$ is the common local payoff of the actions a_i taken by the agent i and a_j taken by the agent j, when agents i and j are connected by the edge (i,j). The quantity $Qi(a_i)$ is the local payoff of the action a_i for the agent i.

During the simulation phase of the MCTS algorithm that runs, the joint action that is taken is obtained by applying the max-plus algorithm to the coordination graph of the particular state of the system.In this state, the agents perform iterative message passing by computing their best response to any possible action a_j from the neighbor agent j, as shown in function 2.8, until convergence or after a predefined amount of time is reached.

$$\mu_{ij}(a_j) = \max_{a_i} \{Q_i(a_i) + Q_{ij}(a_{ij}) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i)\} \tag{2.8}$$

After the graph is solved by using max-plus, the joint action $\overline{A}$ receives a reward $\bar{r}$ and results to a new state $\overline{S}$, as in the traditional MCTS schemes. The reward $\bar{r}$ is computed for every agent independently by using some domain related heuristics and it is used to update the agents statistics. If $\overline{r_i}$ is the reward of the action for the agent i and $\overline{r_j}$ is the reward of the action for the agent j, the statistics of the agents are updated as

$$N_i(\overline{S}, a_i) += 1 \tag{2.9}$$

$$Q_i(\overline{S}, a_i) += \frac{\overline{r_i} - Q_i(\overline{S}, a_i)}{N_i(\overline{S}, a_i)} \tag{2.10}$$

$$N_{ij}(\overline{S}, a_i, a_j) += 1 \tag{2.11}$$

$$Q_{ij}(\overline{S}, a_i, a_j) += \frac{r_e - Q_{ij}(\overline{S}, a_i, a_j)}{N_{ij}(\overline{S}, a_i, a_j)} \tag{2.12}$$

where r_e is the reward of the joint action for i and j and it is computed as $r_e = \overline{r_i} + \overline{r_j}$. $N_i(\overline{S}, a_i)$ and $N_{ij}(\overline{S}, a_i, a_j)$ are the times the actions a_i and the joint action (a_i, a_j) have been selected in previous simulations.

In order to add exploration terms to the scheme, Ghoudhurry,Gupta et al. proposed two distinct phases of computation. The first phase is called **edge exploration** and it adds an exploration term in the final round of the message passing between agents,as in equation 2.13.

$$\mu_{ij}(a_j) = \max_{a_i} \{Q_i(a_i) + Q_{ij}(a_{ij}) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) + c\sqrt{\frac{log(N+1)}{N_{a_i,a_j}}}\} \tag{2.13}$$

The last term provides the exploration component to the system , as per the upper confidence bound (UCB) strategy of the MCTS .This exploration term is applied in the action selection during the expansion of the tree but it is ignored during the final action coordination.

The second phase of exploration is called **node exploration** and it is applied in the final action selection phase of the MCTS. It is a simple modification of the equation 2.7 and it is shown below, in 2.14

$$a_i^* = arg\max_{a_i} \{Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) + c\sqrt{\frac{log(N+1)}{N_{a_i}}}\} \tag{2.14}$$

The aforementioned exploration phases are heuristics in the initial introduction paper of the algorithm.In chapter 3 these heuristics are modified to better match the needs of the lane free traffic planning problem.

In conclusion, the final algorithm as it was presented in [10] is summarized in the algorithmic presentations below.

Algorithm 1 Factored value MCTS with Max-Plus

Input: time limit, depth, c, $\hat{s}$

Initialize N_i, Q_i, N_{ij}, Q_{ij} ▷ Statistics of nodes and edges
function FV-MCTS-MP($\hat{s}$,depth)
  while time limit mot reached **do**
    SIMULATE($\hat{s}, depth$)
  end while
  $\hat{a^\star} \leftarrow$ maxplus(0) ▷ No exploration in the final phase
  return $\hat{a^\star}$
end function

function SIMULATE($\hat{s}$,depth)
  if depth=0 **then**
    return 0
  end if
  $\hat{a} \leftarrow MAXPLUS(c)$
  $\hat{s'}, \hat{r'} \sim T(\hat{s}, \hat{a}), R(\hat{s}, \hat{a})$ ▷ Generative model
  $\hat{q} \leftarrow \hat{r} + \gamma * SIMULATE(\hat{s'}, depth - 1)$
  UPDATESTATS($\hat{s}$,$\hat{a}$,$\hat{q}$)
end function

function UPDATESTATS($\hat{s}$,$\hat{a}$,$\hat{q}$)
  for every agent i **do**
    $N_i(\hat{s}, a_i) += 1$
    $Q_i(\hat{s}, a_i) += \frac{q_i - Q_i(\hat{s}, a_i)}{N_i(\hat{s}, a_i)}$
  end for

  for every edge (i,j) in graph **do**
    $N_{ij}(\hat{s}, a_i, a_j) += 1$
    $q_e \leftarrow q_i + q_j$
    $Q_{ij} += \frac{q_e - Q_{ij}(\hat{s}, a_i, a_j)}{N_{ij}(\hat{s}, a_i, a_j)}$
  end for
end function

Algorithm 2 MaxPlus action selection

Require:
1.Coordination graph $G(s) =< V, E >$
2.State-node statistics N,Q
3.Max iterations M
4.Exploration flag
5.Normalization flag

function MAXPLUS(c)
  for t $\leftarrow$ 1 to M **do**
    $\mu_{a_j} = \mu_{ij} = 0$ for all agents and edges
    for every agent i **do**
      for all neighbors j $\in \Gamma(i)$ **do**
        Compute $m_{ij}(a_j)$ as in Equation 2.8
        if message normalization **then**
          $\mu_{ij}(a_j) -= \frac{1}{|A_j|} \sum_{a_j \in A} m_{ij}(a_{ij})$
        end if
        send message $m_{ij}(a_i)$ to agent j
        if $m_{ij}(a_j)$ close to previous message **then**
          break
        end if
      end for
    end for

    for every agent i **do**
      if edge exploration **then**
        for all neighbors j $\in \Gamma(i)$ **do**
          Compute $m_{ij}(a_j)$ via Equation 2.13
        end for
      end if
      $q_i(a_i) = Q_i(a_i) + \sum_{j \in \Gamma(i)} m_{ij}(a_i)$
      if node exploration **then**
        $Q_i(a_i) += c\sqrt{\frac{log(N+1)}{N_i(a_i)}}$
      end if
      $a_i^{'} = argmax_{a_i} q_i(a_i)$
    end for
    if time limit reached **then**
      break
    end if
  end for
  return $\hat{a'}$
end function

2.2 Related work

2.2.1 Motion planning methods in autonomous driving

One of the fundamental concepts for autonomous driving, is the motion planning of the autonomous vehicles. However, exact solutions for motion planning are computationally intractable [34]. Therefore, there is plenty of proposed variations in literature for motion planning in a vehicular traffic environment. The numerical methods for autonomous vehicles planning in a lane-based traffic setting can be broadly divided in three methods [34], namely Variational methods, Graph-Search methods and Incremental search methods.

Variational methods represent the planning as a high dimensional function representing the quality of a solution, where the parameters represent different aspects of the problem, like the vehicle's velocity, distance to the nearest vehicle etc. These methodologies, solve the motion planning problem by using non-linear optimization techniques over the function [34]. There are many interesting algorithms of this category. One proposed solution uses Euler's method for numerical approximation of the trajectory [50], while another uses finite subsets of the Legendre or Chebyshev polynomials to do the same [11]. The drawback of these methods is that they are usually stack in local minima [34].

Graph-search methods are the methods that represent the vehicle's configuration and environment as a graph, where the vertices represent a collection of vehicles in the route and the vertices represent the interactions between the vehicles. The solution can be extracted then, by applying a search for minimum cost in the graph. Existing methods for constructing the graph involve heuristic methods for algorithmic generation of the graph based on a high level street network map. Other methods use existing geometric methods such as cylindrical algebraic decomposition [23] or constraints on maximum curvature settings [22] in order to create the graph, whereas some methods use sampling in order to explore the reachability of the domain configuration space by generating motion primitives [16] by recording the motion of an expert driver [15]. The main disadvantage of the aforementioned graph-based techniques that search over a fixed graph, is that they search only over the set of paths that can be constructed from motion primitives in the graph.

Incremental search methods is a group of methods that incrementally create a reachability graph (very often as a tree) representing the planning problem in a similar way to the graph-based methods, by sampling the configuration space. When the tree becomes large enough to contain at least one goal region, then the construction of the graph stops and a graph-search algorithm is applied to extract the best possible solution. Because of the vast size of the configuration space, randomized techniques are very often adapted for adding nodes/vertices to the graph. One of the most well-known randomized tree-based incremental planners algorithm is the expansive spaces tree (EST) planning algorithm proposed by Hsu et al. [20]. A technique for finding feasible trajectories for high-dimensional systems is the Rapidly-exploring Random Trees (RRT) [46] . In the RRT method, the exploration is achieved by taking a random sample from the state space and extending the tree in the direction of the sample.

MCTS planning methods

Some of the most common use cases of the incremental search methods are the family of the MCTS algorithms. A very common use case of the MCTS in autonomous

driving problems, is the modelling of the behaviour of the other vehicles. Considering the human intentions in the vehicle behaviour [19], a learning-based continuous MCTS method [27] and an introduction of probabilistic models of the behaviour of the other vehicles that are used in the selection phase of the MCTS controlling [48], are some interesting use cases in that direction .

There are also many interesting endeavours that use the MCTS as part of the autonomous driving environment, performing other tasks than modelling other vehicles, like maneuver prediction with image input [9]. Nevertheless, to the best of our knowledge this is the first work that applies the MCTS for autonomous vehicle planing in a lane-free traffic environment.

2.2.2 Motion planning in Lane-free traffic settings

Under the recently proposed Lane-Free traffic domain, multiple approaches for vehicle movement strategies have already been established. First, a rule-based approach that utilizes the notion of 'forces', i.e., vehicles can be thought of as 'pushing' one another in order to advance, was introduced in [35]. Then, the authors in [30] expanded on this work by improving upon the controller design, and providing a more extended evaluation and insights in that direction.

Besides rule-based approaches, there are strategies designed based on optimization and learning. Specifically, authors in [49] introduce an optimal control approach for the problem, utilizing the model of vehicle kinematics and considering also obstacles like the road boundaries, in order to optimize for a future horizon. The optimization criterion incorporates sub-objectives for efficient vehicular travelling and collision avoidance [49]. Another approach tackles the problem with deep reinforcement learning algorithms to the domain, making the vehicles improve their decision-making directly from observed experience [25]. Finally, gaining a different perspective, the problem of autonomous driving as a multi-agent collaborative environment has been addressed in [18] by adopting an approach that models the problem as a coordination graph and searches for an appropriate solution in the graph using the Max-plus algorithm.

Compared to these approaches, our work in this thesis can be seen as another slightly different approach to the motion planning in lane-free traffic settings domain, that takes advantage of the anytime nature of the MCTS, something that the aforementioned algorithms don't do. A comparison among these solutions and this work, studying the advantages and disadvantages of each approach in the domain, is outside the scope of this work but could lead to an interesting direction for future work.

Chapter 3

Our approach

In this section we present our approach, which can be distinguished in two phasese. First, we *formulate* the problem of lane-free driving considering the MCTS algorithm, where every vehicle executes the algorithm independently, and other vehicles are observed as moving obstacles. Then, we tackle the multi-agent nature of a road traffic environment with the FV-MCTS algorithm [10], where we adopt our formulation appropriately in order to utilize communication among vehicles as well.

3.1 Monte Carlo Tree Search for Lane-Free Driving

3.1.1 State and Action Space

In order to apply the MCTS algorithm in a specific problem, there two fundamental sets that must be defined, the state space S and the action space A. In our approach every vehicle adopts the algorithm in order to take an action, independently and separately of the other vehicles. Therefore, in a given time t the state for a vehicle c comprises of its position, velocity and desired velocity, as well as the positions and velocities of all the vehicles that are in a longitudinal distance smaller than d from the main vehicle that runs the algorithm. Formally, we can define a vehicle c with information about its position and dimensions as a set $c = \{p_x, p_y, v_x, v_y, l, w, d_v\}$ where the members of the set are described in Table 3.1.

p_x	Position in x-axis
p_y	Position in y-axis
v_x	Velocity in x-axis
v_y	Velocity in y-axis
l	Vehicle length
w	Vehicle width
d_v	Desired speed

TABLE 3.1: A vehicle's definition

Having the vehicle defined, a state s can be defined as a set $s = \{c_m, \Gamma = [c_1 ... c_n]\}$ where c_m is the ego vehicle and the rows of the matrix Γ consists of the vehicles that are close to the main vehicle. A terminal state, finally, is a state where the main car is crushed with another car or has exceeded the road boundaries.

An action that a vehicle can take in our setting consists of both the longitudinal and lateral accelerations that can be applied to it at a given time t. In theory, a vehicle c can take any action $\alpha = \{a_x, a_y\}$ with a_x being the longitudinal acceleration and a_y being the lateral acceleration, ranging from $-a_l(\frac{m}{s^2})$ to $+a_h(\frac{m}{s^2})$, where a_l and a_h are real numbers depending on the vehicles characteristics. These acceleration

quantities can take any real number. Intuitively, when the a_x is negative actually means deceleration and a positive value of a_y actually means turning left whereas a negative means turning right.

Nevertheless, the action space A must be discrete for a search tree setting, therefore we defined the action space using only a small discrete number of 7 possible accelerations for the longitudinal acceleration and 3 for the lateral. Finally, an action α is defined as a set $\alpha = \{\alpha_x, \alpha_y\}$ of accelerations, where α_x is the longitudinal and α_y is the lateral acceleration. The quantities α_x and α_y can only take specific values. The configuration of the values that they can take is shown: in Table 3.2.

Longitudinal Acceleration Values	Lateral Acceleration Values
$\alpha_x(m/s^2) = \{-5, -4, -3, -1, 1, 2, 4, 5, 6\}$	$\alpha_y(m/s^2) = \{-2, -1, 0, 1, 2\}$

TABLE 3.2: Configuration of the possible acceleration values that can be applied to a vehicle.

The action space A consists of all the possible combinations of longitudinal and lateral acceleration values, therefore there are 21 possible actions in this space.

3.1.2 Reward functions

The goal of the vehicles is to avoid collisions and to travel with their desired speed whenever possible. Therefore, the heuristic reward of a state that a simulation gives should depend on: 1) the likelihood for collisions and 2)the proximity to the desired speed. One simulation in our approach stops when the simulation gets in a terminal state or after n forward steps.

For the first objective, the heuristic adapts the proposed artificial potential fields [18] to promote collision avoidance, together with a negative value for actual collisions occurred during the simulation. Essentially, when a collision between the main vehicle and another vehicle occurs in the simulation, a terminal state is reached before n timestamps. The deeper in one simulation the collision happens, the lower the negative score at the current state. Adding a negative value when a collision happens in a simulation seems reasonable because this means that a collision is more likely to occur in that state. However, an action can be a bad action when it leads the vehicle to a state with higher probability of collision, even if the collisions in the simulations starting from this state are relatively few. To give a negative value in these actions in order to affect collision avoidance, the aforementioned artificial potential fields are taken into consideration. These fields consider the positions and velocities of two vehicles and give a quantification f_{ij} of the potential collision between these two vehicles. In our proposal, the pairs between all neighboring vehicles and the main vehicles in the final simulation state are accumulated to give a total 'potential collision' of a simulation, only if the two vehicles are closer than d.

Finally, the part of the total reward of a simulation that tackles collision avoidance is given in Equation 3.1

$$g(s) = \begin{cases} \frac{D}{n_s} + \sum_{v=1}^{M} f_{i,v} & \text{if } dist(i,v) < d \text{ and } n_s < n \\ \sum_{v=1}^{M} f_{i,v} & \text{if } dist(i,v) < d \text{ and } n_s \geq n \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where:

- D: A positive constant

- n_s: The number of steps before collision. The term $\frac{D}{n_s}$ in case of a collision occurrence quantifies the significance of the collision depending on how much time we have before it occurs
- M: The number of vehicles of $dist(i, v) < MaxD$, for vehicles $i, v$and a constant $MaxD$
- $dist(i, v)$: The longitudinal distance between two vehicles i and v
- $f_{i,v}$: The quantified potential collision between two vehicles i, v according to artificial potential fields
- n: The number of steps in a simulation

For the second objective, there is no actual uncertainty about the outcome of an action to be inferred by Monte Carlo simulations. When a vehicle is at a state and takes an action, the velocity of the next state is a deterministic outcome. It is determined by the kinematic equation $v_{t+1} = v_t + a_t T$. The position of the vehicle is also updated in a deterministic manner by $x_{t+1} = x_t + v_t T + \frac{1}{2} a_t T^2$, where x_t is the position of the vehicle at the time t, v_t is the velocity and a_t is the applied acceleration of the vehicle at the time t. T is a constant time period. Given the deterministic nature of the outcome of an action, the score of a state is regardless of the simulation returns. The difference between the longitudinal velocity and the desired speed of a vehicle is considered in order to maintain the avoid collisions-reach the desired speed trade-off. The part of the score of a simulation regarding the minimization of the difference between the actual vehicle velocity $v_t(s)$ at the state s and the desired vehicle velocity v_d is shown in Equation 3.2

$$f(s) = \frac{\varepsilon}{|v_t(s) - v_d| + \varepsilon} \tag{3.2}$$

In order to consider both objectives, the total reward of a simulation is calculated as a weighted sum:

$$G(S) = A * g(s) + B * f(s), \text{A} \geq 0, \text{B} \geq 0 \tag{3.3}$$

The trade-off between collision avoidance and retaining the desired speed is addressed by the coefficients A and B. For example, when A=0 than the algorithm does not provide collision avoidance at all and if it is a bigger number than the algorithm mainly focuses in collision avoidance rather than the vehicle's desired speed. Both A and B are heuristics that can be adjusted during simulation.

3.1.3 The tree construction

So far, the state space, action space and the reward functions have been defined, but not the search tree itself. In order to actually define the tree for the domain in hand, the node of a tree has to be defined.

The node of a tree in our proposal consists of a state, an action that leads to that state and the possible actions (which in our case is the entire action space for every possible state without restrictions) that the vehicle can take from this state and can expand the tree by creating new nodes. Moreover, the reward function G(s) from Equation 3.3 that defines the score of a single simulation is used and the average score of the rewards of the simulations started from this node or any of its children nodes.

Algorithm 3 A High Level MCTS implementation

```
t ← 0
root ← initTree()
while t ≤ T do
    s ← selectNode(root)
    if isTerminal(s) then
        continue
    end if
    if numOfVisits(s) ≥ N then
        s ← expandNode(s)
    end if
    score ← simulate(s)
    backPropagate(s,score)
    t ← t + 1
end while
```

The *expandNode* and the *backPropagate* functions from Algorithm 3 showcase the basic MCTS, node expansion and score backpropagation actions, as shown in Figure 2.1. In our work, the node is expanded by taking the possible actions following a specific order. If the node is expanded to have a children node from an action a_i, the next expansion comes from the action a_{i+1} etc. While these functions are domain agnostic, the tree policy and the simulation policy implementations can differ in applications.

3.1.4 Selection policies and simulation policies

In Algorithm 3 we invoke the selectNode and simulate functions that represent the tree and the simulation policy that our algorithm follows. In this subsection we elaborate on these policies.

One node is selected in our proposal, based on the UCT formula Equation 2.3 analyzed in Section 2.1.1. Every node keeps a score from the previous simulations and the average of this score is the $Q^{\star}(s,a)$ of the aforementioned equation.

The simulation policy we use goes up to M steps ahead of the current state at time t, and does not search until a terminal state is found. The rewards of a simulation are as described in Section 3.1.2. The actions that a vehicle takes at each intermediate state in the simulations are sampled from a uniform distribution from the action space. This way, we ensure that many different possible sequential actions are taken in the simulation. After an action is executed, the agent simulates a new state where other vehicles have moved as well. Since it is a simulation and the algorithm does not know the actions of the other vehicles, estimations have to be made about their updated positions and velocities at every simulation state. The assumption that we do is that no acceleration has applied to the other vehicles in any direction. As such, the velocity in each direction is constant and the position of the vehicle is updated by $x_{i+1} = x_i + v_i * T$, for every intermediate state.

Having computed the positions and velocities of the vehicles at each simulation state, the simulation reward is again computed by the formulas described in the Section 3.1.2.

3.2 FV-MCTS using Max-Plus

For a multi-agent approach to the lane-free vehicle planning with MCTS, we propose the adoption of the approach in [10], as presented in Section 2.1.2. Our environment is now a collaborative multiagent environment and the MCTS planning is no longer applied independently by the agents. Instead, each agent takes an action based on the score it receives from applying the Max-plus algorithm to the coordination graph, considering itself as the ego vehicle. In order to adapt this collaborative environment to the algorithm, there are some additional parameters that have to be specified, referring to the coordination graph, and a different formulation; different definitions of an action and a state for the MCTS.

3.2.1 State and action space

In this approach, in contrast with the previously proposed plain MCTS algorithm, a state at a particular time t is a global state and it is common between all the agents. It is called the 'joint state' and it consists of the positions and velocities of all the vehicles at this timestamp and the coordination graph that these vehicles form. Therefore, the joint state, which would represent a node in the tree, as can be seen later, takes values from the joint state space, which is the combination of all the state spaces of the agents as introduced in Equation 3.4.

$$S^* = S_1 \times S_2 \times ... \times S_n \tag{3.4}$$

Similarly, since we consider a global state, a notion of global joint action space has to be introduced. This joint action space consists of the combination of the actions that each agents can take in a given state as shown in Equation 3.5.

$$A^* = A_1 \times A_2 \times ... \times A_n \tag{3.5}$$

The action space A_i of a vehicle i, is a tuple of accelerations as described in Section 3.1.1. The system takes an action from this action space collaboratively and than for each individual vehicle v , applies the individual action of v participating to the taken global action, to the corresponding vehicle v. The aforementioned procedure is different from the plain MCTS where each agent takes action independently.

3.2.2 Coordination graph

Following the concept of FV-MCTS [10], we need to construct a coordination graph where nodes are the agents interacting with each other according to the corresponding edges. We define a maximimum distance D between agents as a threshold to decide whether to create an edge between them. If the distance between them in a given state is larger than D, then there is no edge between them in the graph.

The algorithm keeps statistics for the joint actions that have been previously taken and applies message passing using the most recent statistics, as in Equation 2.5. After a number r of iterative message passing rounds, the process terminates and the joint action is determined. Some technical hyperparameters are adjusted for the domain of planning vehicle trajectories for lane-free traffic environment, which are discussed in more detail in Section 4.3. The main algorithm, however, that we applied to the problem is exactly as the one described in Section 2.1.3.

3.2.3 Adaptation to lane-free traffic planning

In line 13 of Algorithm 2 in Section 2.1.3, the next state and the immediate rewards of one step are based on the problem at hand. In our case, the next state $\bar{s}$ is simply determined by the positions and velocities of the vehicles in that state and the coordination graph they form. Calculating the velocities and the position for each vehicle i , we get the next states $\overline{s_i}$ of the vehicle, in the same way as in Section 3.1.1. After that, we compute the coordination graph based on the distances between vehicles in the new states $\overline{s_i}$. Finally, the joint next state $\bar{s}$ is the combination of the states $\overline{s_i}$ and the coordination graph, as it is described in Section 3.2.1.

The immediate reward $\bar{r}$ is computed by the following Equation 3.6:

$$\bar{r} = A * f(\bar{s}) - B * g(\bar{s}) \tag{3.6}$$

where the constants A and B are always positive and determine the attention of the system towards reaching the desired speed for the vehicles or collision avoidance between the vehicles. Therefore, the algorithm seeks increasing the quantity $f(\bar{s})$ and decreasing the quantity $g(\bar{s})$ by parametrizing the constants A and B.

The quantity $f(\bar{s})$ computes the velocity proximity of the vehicles in the state $\bar{s}$ to their desired speeds. In our proposal, it is computed as it is shown in Equation 3.7 and it is the weighted summation of the differences between actual speeds and desired speeds for all the vehicles in the graph G of the state $\bar{s}$. It can be inferred from 3.7 that the lower the difference from the desired speed for the vehicles, the larger the quantity $f(\bar{s})$ becomes:

$$f(\bar{s}) = \sum_{i \in G} \frac{X}{|v_{x_i} - v_{d_i}| + e_1} \tag{3.7}$$

The second quantity $g(\bar{s})$ addresses collision avoidance between vehicles. For every vehicle, it accumulates a quantity that depends on the positional distance in both axes x and y, between the vehicle and the neighboring vehicles $\Gamma(i)$ connected with it in the graph, and sums for all vehicles in the graph G. It takes positive values that approach zero when the possibility of collision decreases and large values when the probability of collision for the vehicle i is high. The Equation 3.8 shows in detail how it is computed in our proposed solution

$$g(\bar{s}) = \sum_{i \in G} \sum_{j \in \Gamma(i)} \frac{C}{|x_{\bar{s}_i} - x_{\bar{s}_j}| + e_2} + \frac{D}{|y_{\bar{s}_i} - y_{\bar{s}_j}| + e_3} \tag{3.8}$$

The difference $|x_{\bar{s}_i} - x_{\bar{s}_j}|$ is the distance in the longitudinal axis between two vehicles connected in the graph and the quantity $|y_{\bar{s}_i} - y_{\bar{s}_j}|$ is the lateral position difference. The larger the distance between vehicles, the lower this quantity becomes. This leads to lower negative impact in the reward function 3.6. Constants C and D can be parameterized to weigh one distance metric over the other. Constants e_1, e_2 and e_3 are added to the denominators with very small values ($\approx 10^{-10}$) in order to avoid division by zero.

Chapter 4

Experimental evaluation

To evaluate our methods, we conducted all experiments using an extension of SUMO that is suitable for simulating lane-free vehicular traffic scenarios. The TrafficFluid-Sim simulator [45] is build on top of the SUMO software and provides, alongside with all the standard SUMO capabilities, a simulation environment that enables lane-free vehicle movement, i.e., the vehicles can take any lateral position and 'apply nudging' from one vehicle to another. These are two important characteristic of the simulator that make it suitable for a lane-free traffic environment.

Using the TrafficFluid-Sim simulator for connected and automated vehicles, several experiments were run to evaluate the proposed approach. In Subsection 4.1, we present the parameters of the simulation scenarios that are common across all the experiments. The experiments can be separated in two distinct parts. First, the conventional Monte Carlo Tree Search approach that we discussed in Subsection 3.1 was implemented and thoroughly examined. The results of these experiments are presented in Subsection 4.2, along with a related discussion.

In the second part of the experiments, we examined the extension of the aforementioned implementation of the Factored Value MCTS with max-plus [10]. The results of this simulations are provided in Subsection 4.3, where we examine specific coordination scenarios and compare the performance with that of MCTS.

4.1 Experiment parameters

To evaluate our approach, most parameters have consistent values for all experiments, while some have different values depending on the examined scenario, e.g., the duration of the simulation. In Table 4.1, the related parameters that are common for all the experiments are shown.

Parameter	Value
v_d range	$[25, 35] m/s$
time-interval	$0.25s$
highway length	$1000m$
highway width	$10.2m$
vehicle length	$3.2m$
vehicle width	$2m$

TABLE 4.1: The simulations parameters

All the experiments are conducted in a straight road of a fixed width. Furthermore, to compare the different algorithms we consider vehicles with desired speeds coming from the same uniform distribution.

The desired speed for each vehicle takes random values ranging from $25\frac{m}{s}(=90\frac{km}{h})$ to $35\frac{m}{s}(=126\frac{km}{h})$ and is sampled from a uniform distribution. Each vehicle enters the highway at at the beginning of the highway ($p_x = 0$) and a random lateral position within the boundaries of the highway, according to a demand handling method that spawns the vehicles laterally randomly, but also considering nearby traffic, i.e., a vehicle will not spawn behind another one causing dangerous and unstable traffic situations. For more information, we refer the reader to [18].

4.2 MCTS performance

To evaluate our algorithm, we keep some fixed values for the simulation and alter, in every experiment, the velocity and lateral position with which the different vehicles enter the highway, as well as the traffic flow. The road and vehicle dimensions remain consistent for every simulation.The time interval of the SUMO also is kept stable for every individual experiment. Table 4.1 shows the specific values of these parameters for the simulations.

4.2.1 Discrete action space

The possible actions that a vehicle can take are 21; all the possible combinations between the 7 longitudinal and 3 lateral acceleration values. The aforementioned acceleration values are presented in Table 4.2.

Longitudinal Acceleration Values	Lateral Acceleration Values
$\alpha_x(m/s^2) = \{-5, -2, -1, 0, 1, 2, 5\}$	$\alpha_y(m/s^2) = \{-1, 0, 1\}$

TABLE 4.2: The possible actions for the experiments are the 21 combinations between the longitudinal and lateral acceleration values

4.2.2 Simulations parameters

To evaluate the performance of the MCTS implementation, 8 different scenarios were set and run with the SUMO simulator using the TrafficFluid-Sim simulator discussed at the beginning of this chapter.

The seed for the pseudorandom generators that generate the desired speed and the lateral entry position of each vehicle was common in all these simulations. Each experiment was examined with two different configurations of entry longitudinal speed. In the first configuration, every vehicle enters the road with a longitudinal speed of $20\frac{m}{s}$ and in the second configuration every vehicle enters the road with a longitudinal speed of $25\frac{m}{s}$. All vehicles in every simulation are lane-free vehicles that take actions according to our approach.

Every simulation is $3600sec$ (1 Hour) long, with a varying demand flow at the entry point of the highway. More specifically, the different flows for the different experiments are shown in Table 4.3. It is important to mention that the flows selected exceed the maximum flow of lane-based traffic. Note that we experiment on a 10.2 meter road which corresponds to a 3-lane conventional highway scenario. In an equivalent lane-based scenario with 3 lanes, the maximum flow would be approximately $7300veh/hr$.

Simulation	1	2	3	4	5	6	7	8
Flow(Veh/H)	5500	7000	8000	8400	8800	9200	9600	11000

TABLE 4.3: The different traffic flows for the simulations of the MCTS experiments

4.2.3 MCTS performance evaluation

The two objectives that are of interest to us are the collisions that occur during a simulation, showing the performance of our approach regarding safety, along with the average delay of vehicles, showing the effectiveness of the algorithm.

Collision Occurrences

For every simulation, the total number of collisions occurred is measured. The results of every simulation are shown in Figure 4.1.

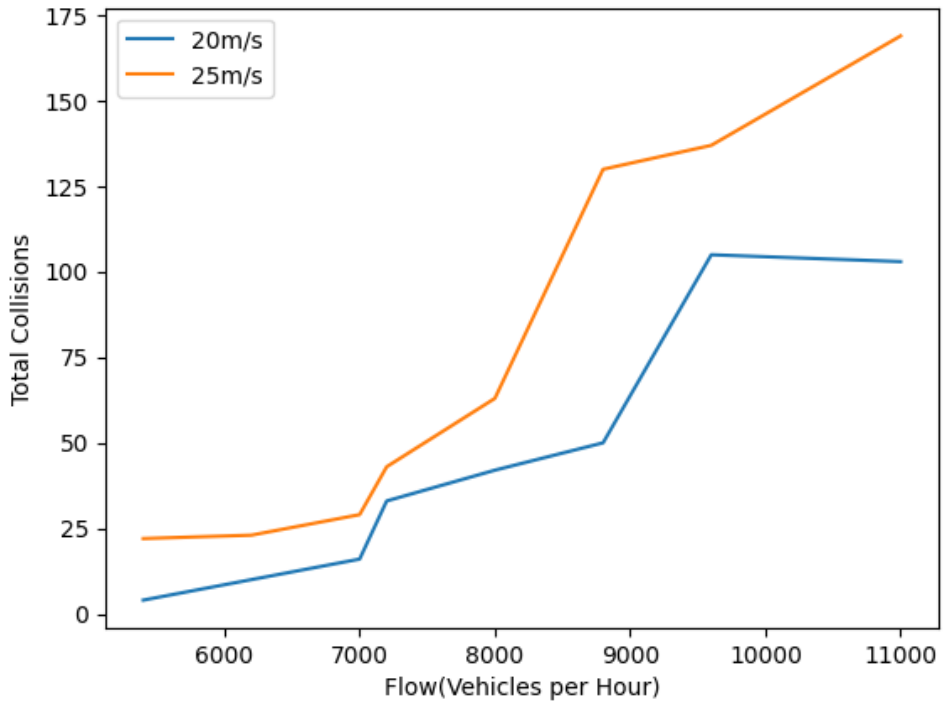


FIGURE 4.1: Collisions per timestamp for all cars combined

The two different plot lines correspond to the entry speed for all vehicles. As expected, a higher entry speed causes more collision. It can also be seen that the increase of collision occurrence is more abrupt with the increase of traffic flow, in the area between $7000 veh/hr$ and $9500 veh/hr$ in the figure, whereas it is almost constant elsewhere. The flatness of the lines at the very right of the figure can be explained by the fact that our library cannot really simulate such high traffic flows given the demand related parameters that address safety. In reality, SUMO extends the simulation time in cases where the number of vehicles about to enter a highway is very high for a time period, therefore the actual traffic flow of the simulation is not that high.

Putting this aside, it can be seen that the vehicles following the algorithm are causing very few collisions when the flow is in the range of a lane-based scenario with three lanes, below $7300 Veh/hr$. This shows that the MCTS algorithm is effective regarding safety in reasonable vehicular traffic scenarios. In very high traffic flows, however, there is an increase of collisions, although the total number remains relatively low when considering the total number of interactions within the time period of 1 hour. This shows the limitations of the algorithm in lane-free environments where the traffic flow can be extremely high and there is no sense of structure that that road lanes can provide.

Average deviation from the desired speed

At every simulation timestamp, the difference between the actual longitudinal speed and the desired speed of each vehicle is measured. The average of the above quantity of all the n vehicles in the road at this timestamp, is the average deviation from the desired speed at this particular timestamp and it is shown in Equation 4.1.

$$v_{diff}(t) = \frac{1}{n} \sum_{i=1}^{n} |v_{d_i}(t) - v_i(t)| \tag{4.1}$$

where $v_{d_i}(t)$ is the desired speed and $v_i(t)$ is the actual speed of the vehicle i at the time t.

The average of the quantity $v_{diff}(t)$ of the Equation 4.1 of all the simulation timestamps, is the total deviation from the desired speed. This quanity shows whether the vehicles travel according to their desires or compromise their speed for achieving safety.

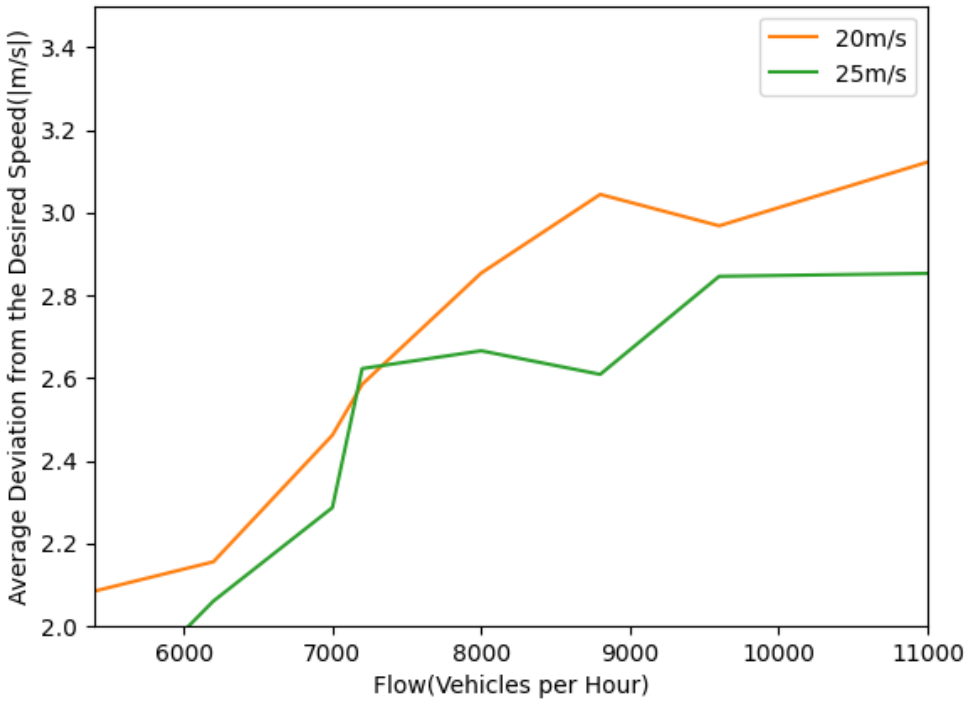


FIGURE 4.2: Average deviation from the desired speed, for different flows and vehicular entering speeds

The correlation between the two lines indicates that the average deviation from the desired speed is in general higher for the case where the vehicles enter the highway with an initial speed equal to 20m/s, compare to the case where they enter the highway with an initial speed of 25m/s. This was expected because the deviation from the desired speed is higher at the early timestamps when a vehicle enters the highway with 20m/s compared to when it enters with a speed of 25m/s, since the desired speed for any vehicle ranges between 25m/s and 35m/s. The fact that the orange line is above the green line in Figure 4.2 is therefore of no interest here.

However, one interesting result to notice from the Figure 4.2 is the fact that the average deviation from the desired speed is increasing slowly in the increase of the traffic flow and does not get above |3m/s| in any case. This shows that the algorithm can lead the vehicles to take actions that can make them achieve their desired speeds, even when the traffic flow is high and it is more difficult to do so without compromising safety.

Another interesting thing to consider is the performance of the algorithm regarding the average deviation from the desired speed over time. MCTS is an algorithm that delivers results from the very early stages of the simulation, since at each timestamp of the simulation, only the current timestamp is considered to create the states for the search tree and no type of learning or modelling of other vehicles' future behaviour is involved. The average deviation from the desired for various flows on time, is illustrated in Figure 4.3.

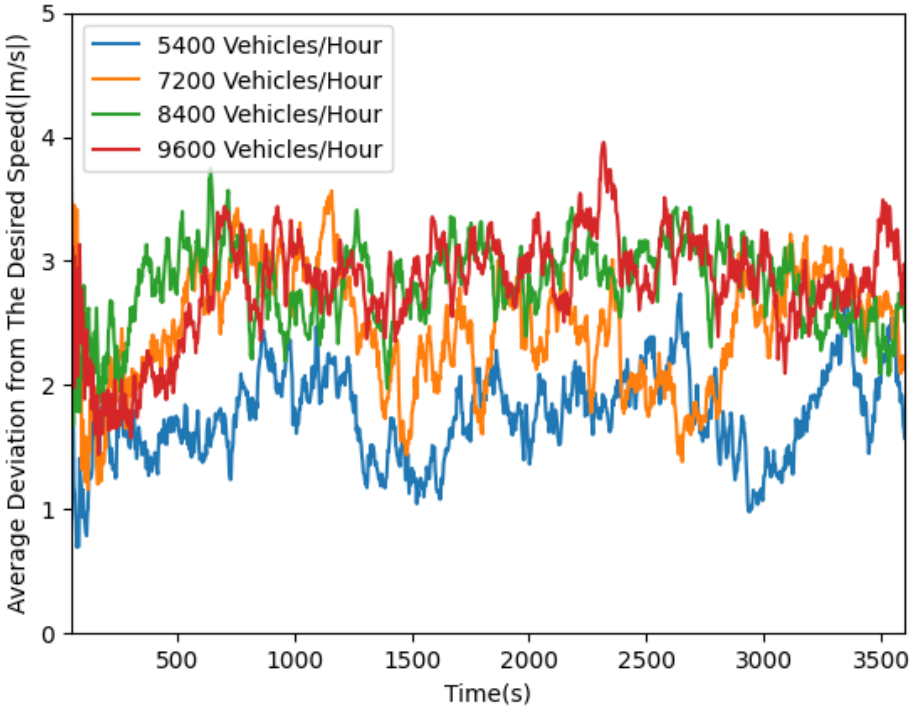
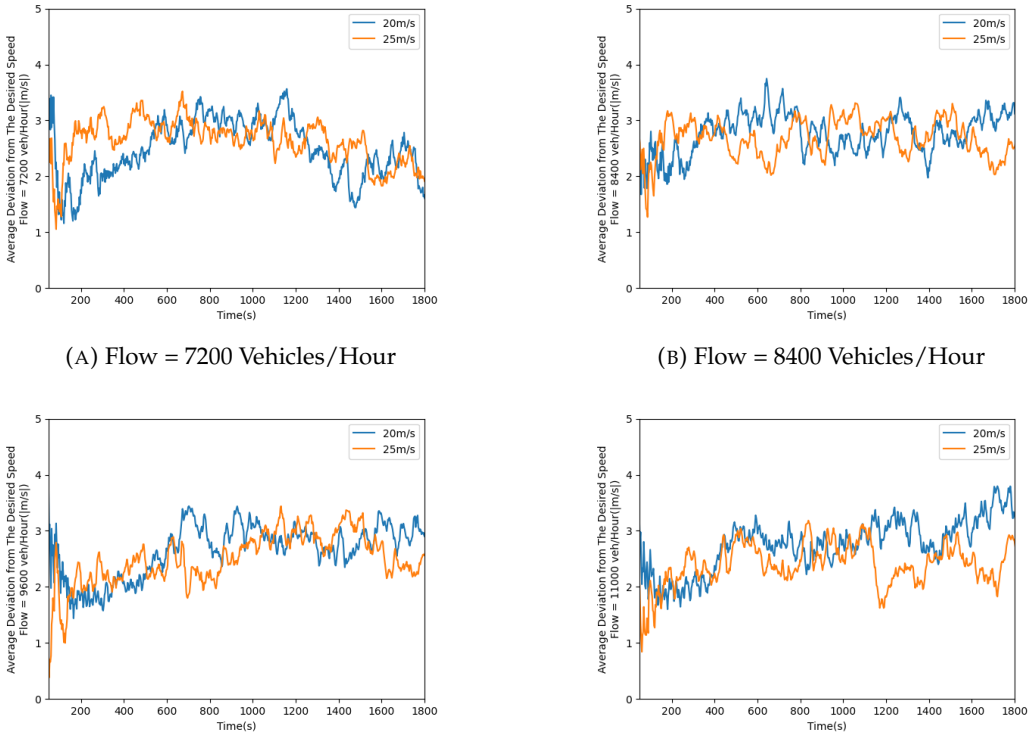


FIGURE 4.3: Average deviation from the desired speed on different time stamps for vehicles entering highway with a speed of 20m/s

Putting aside the expected result that the lines of the simulations for lower traffic flows result in lower deviation from the desired speed, since lower traffic flow means fewer obstacles for the vehicles, there is one interesting inference from the Figure 4.3; even at the early stages of the simulation, the algorithm delivers the same results. There is no correlation between the time and the average deviation from the desired speed, which justifies our premise that MCTS delivers results from the very early steps of the simulation.

Finally, the velocity with which the vehicles enter the highway does not correlate with the average deviation from the desired speed regarding time, as it can be seen in Figure 4.4.



(A) Flow = 7200 Vehicles/Hour

(B) Flow = 8400 Vehicles/Hour

(C) Flow = 9600 Vehicles/Hour

(D) Flow = 11000 Vehicles/Hour

FIGURE 4.4: Average deviation from the desired speed at each time stamp

The common behaviour of the simulation results in 4.4 is that the average deviation from the desired speed does not significantly change with respect to time, but changes without a pattern and remains inside a specific range. This behaviour is common for both cases, depending on the entrance speed.

Average delay

The delay of a vehicle is the difference between the time a vehicle should have taken from the starting of the highway to the ending of the highway if it was travelling with the desired speed velocity v_d, and the time that the vehicle actually did to complete its route in the simulation.

Considering that the highway of the experiments is a straight highway without turning points, the desired time t_d can be computed from the length of the highway r_l and the desired speed of the vehicle v_d as $t_d = \frac{r_l}{v_d}$. Now, if we consider the time t_v that the vehicle did to get from the starting of the highway to the ending of it in a simulation, the delay of this vehicle is $d = t_d - t_v$. The average delay is the average of the aforementioned quantity for all vehicles.

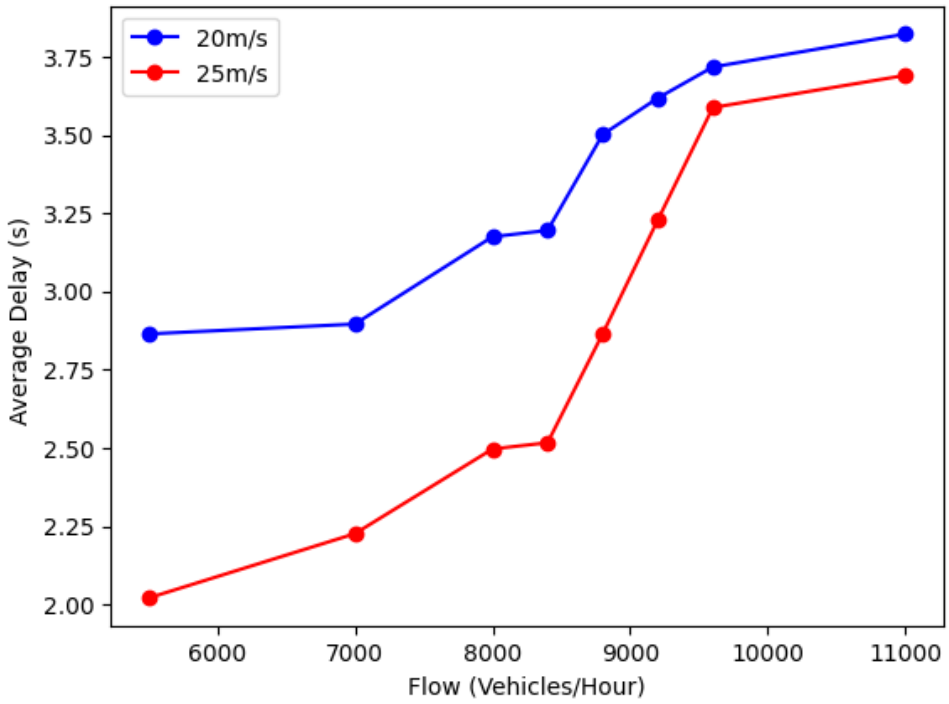


FIGURE 4.5: Average delay for different traffic flows and vehicular entering speed

As it can be seen from Figure 4.5, the average delay increases, when the traffic flow increases. This happens because a very high traffic flow means that the vehicle has many other vehicles surrounding, therefore it cannot surpass vehicles with lower desired speed easily. The delay becomes higher more rapidly in flows higher than $8000 Veh/hr$ for both vehicular entry speed cases. This indicates a saturation in the algorithm's performance that causes the vehicles to travel relatively slower to avoid collisions in these traffic flow scenarios. This behaviour is caused by two facts, according to our empirical observations. First, the fact that the available actions for each vehicle is limited, therefore it is difficult for the vehicles to surpass other vehicles in high traffic scenarios. Second, the absence of collaboration between vehicles, that could potentially lead to smarter decisions for the benefit of the system as a whole. In this MCTS implementation, there is no communication between vehicles that could help them achieve their desired speeds with collaborate rather than the one potentially disrupting the other's behavior due to its own goal.

4.3 Factored Value MCTS with Max-plus

Computational limitations made the experiments with many vehicles insufficient for the Factored value MCTS with max-plus scenario, as can be seen in Subsection 4.3.2. This is the reason why we did not perform extended experiments with a demand of thousands of vehicles in a reasonable amount of time, like in the experiments for the MCTS case. Instead, we examined three different scenarios involving a small number of vehicles in which we show the multi-agent approach performs better

than the plain MCTS approach without vehicle collaboration. FV-MCTS enables collaborative behavior among the vehicles through the combination of the max-plus algorithm and the MCTS approach.

In the first subsection of this section, the experiment hyperparameters common for all the experiments of the FV-MCTS with max-plus case are presented. Then, in Subsection 4.3.2, the intractability of the second FV-MCTS with max-plus algorithm is shown, when implemented in a single-thread without parallelization, as it has been the case in our implementation. Finally, in Subsection 4.3.3 the three different scenarios are presented in detail, with a related discussion of results for each scenario.

4.3.1 Hyperparameter tuning for FV-MCTS

The hyperparameters that were used in our experiments are shown in the table below.

Minimum distance for creating edge	35m
Maximum number of edges for node	3
Edge exploration constant C	0.5
Max tree depth	3
Simulations from root	12
Max-plus message passing rounds	2
A	10
B	50
X	9.5
C	20
D	10
Safety Gap in road boundaries	1m

TABLE 4.4: The hyper-parameters of the multi-agent implementation

The first six parameters in the Table 4.4 are referring to the FV-MCTS algorithm itself, as it was described in Subsection 2.1.3. The parameters A, B, X, C and D refer to the adaptation of the Factored value MCTS algorithm to our domain that is described in Subsection 3.2. The last parameter, refers to the minimum distance inside the road, from the vehicle center to the road boundary, where the algorithm considers the vehicles to be inside of it. If a vehicle is closer than this quantity to the road boundary, then the algorithm considers that this vehicle is outside the road. This prevents the vehicles from going very close to the road boundaries, in the expense that the road capacity slightly decreases. The behaviour of this trade-off is outside the scope of this work. Finally, all the values of the parameters in Table 4.4 have been extracted after empirical evaluation of different tunings on the 3 experiments.

In contrast to the experiments of the plain MCTS case, here only five individual actions are considered, which are either towards longitudinal or lateral acceleration and not both, as in the plain MCTS case. In order to properly compare the performance of this algorithm to the plain MCTS case, the experiments of the MCTS algorithm on these scenarios use the same action space with 5 actions. The reduced action space has the acceleration values appearing in Table 4.5.

Acceleration Values	Direction
-2 $\frac{m}{s^2}$	Longitudinal
+2 $\frac{m}{s^2}$	Longitudinal
0	
-1 $\frac{m}{s^2}$	Lateral
+1 $\frac{m}{s^2}$	Lateral

TABLE 4.5: The possible actions for the experiments are the 5 cases of either longitudinal or lateral acceleration

4.3.2 Computational performance comparison with MCTS

In order to quantify the fact that performing extended simulations for the FV-MCTS with max-plus implementation is time consuming, therefore impractical at this point, we exhibit the time performance stemming from one experiment populated with multiple vehicles.

For a straight route of 1000m with 50 vehicles spreading uniformly inside it, the average computational time of one timestamp (0.25sec. of simulation time) was computed, for the timeframes when all the 50 vehicles were in the road. The results in Table 4.6 show that an extended experiment of 3600 seconds in simulation time (14000 timestamps in a 0.25s time step), like the simulations of the MCTS algorithm, would take approximately 17 days, therefore rendering the FV-MCTS approach inapplicable for large simulations.

MCTS	FV-MCTS with max-plus
450ms	2m 34s

TABLE 4.6: The average decision time for the system in one simulation timestamp when populated with 50 vehicles

4.3.3 Experiments on 3 Scenarios with the FV-MCTS algorithm

Scenario 1

In this scenario, there are three vehicles that need to collaborate in order to achieve their desired speed all of them. There is one slow vehicle with desired speed 28m/s in front of a vehicle with desired speed of 35m/s that cannot surpass the first one, because there is the road boundary at the left and another slower vehicle at the right of it. An illustration of the initial vehicular positions of the particular example is shown in Figure 4.6.

In the plain MCTS scenario without collaboration, the front vehicle (Car1) and the vehicle in the right (Car2), do not anticipate the intentions of the third vehicle (Car3), since they have reached their individual goals. Figures 4.7a and 4.8a show the trajectories of the three vehicles when MCTS is applied for both longitudinal and lateral positions p_x, p_y respectively, both with respect to time (in seconds).

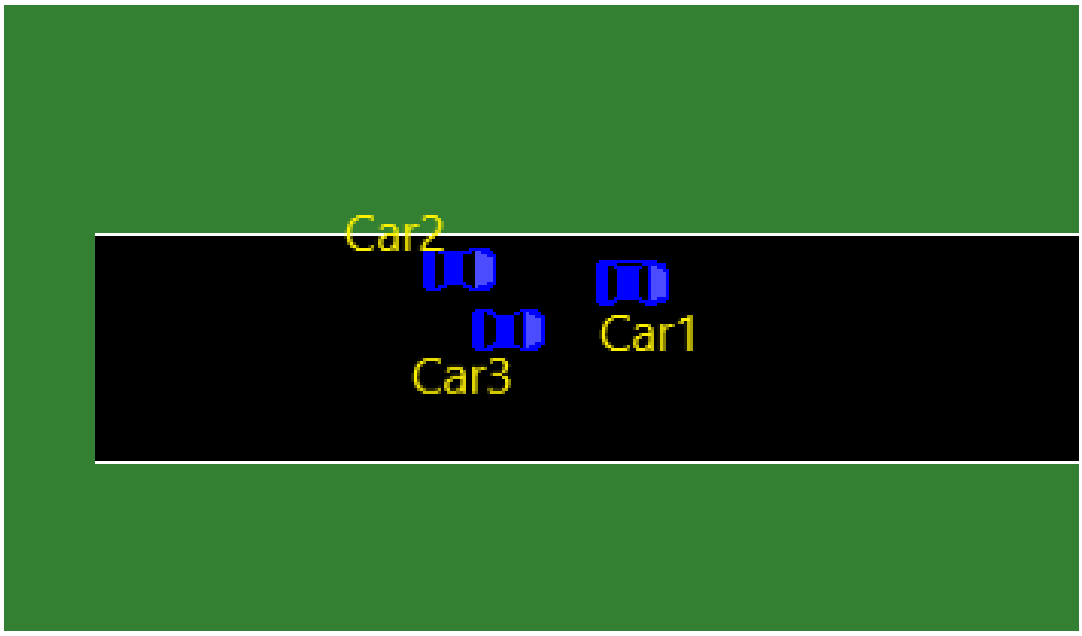
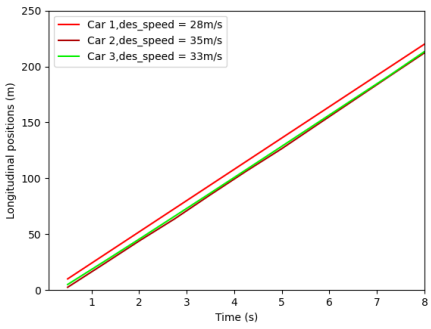
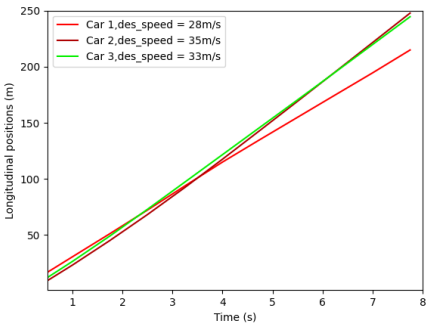


FIGURE 4.6: Initial positions of the three vehicles

As it can be seen from Figure 4.7a, Car2 cannot surpass Car1, although it has higher desired speed. This is because it does not have enough space, as can be seen in Figure 4.8a. Therefore, it just stays behind car1 since there is no collaboration, while car1 is already travelling with its desired speed and has no reason to either move laterally or apply any acceleration, as that would not increase its rewards. To further investigate the algorithm's behaviour in this experiment, Figure 4.9a shows the longitudinal accelerations of the vehicles on time, where it can be seen that Car1 struggles to reach its desired speed. Furthermore, Car2 does not accelerate because it "sees" Car1 and tries to first move on the right and then apply acceleration. Figure 4.10a shows the lateral accelerations of the vehicles at any given time for the plain MCTS case, where the aforementioned behaviour is illustrated.

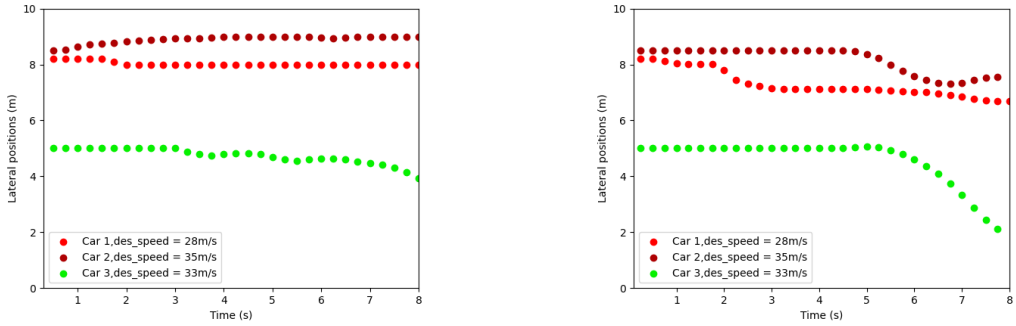


(A) Vehicles longitudinal positions for MCTS case



(B) Vehicles longitudinal positions for Multiagent case

FIGURE 4.7: Longitudinal positions of the vehicles on the road with respect to time



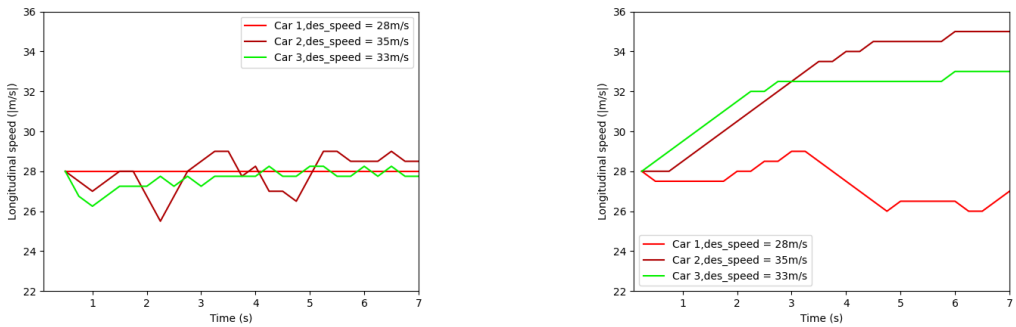
(A) Vehicles lateral positions for the plain MCTS case

(B) Vehicles lateral positions for the Multiagent case

FIGURE 4.8: Lateral positions of the vehicles on the road with respect to time

It can be easily inferred from this experiment that the plain MCTS algorithm has some disadvantages when used for lane-free vehicular planning in a collaborative manner; it cannot find the optimal solution for all the vehicles when the desired state of one vehicle counters, even temporary, the desired state of another vehicle. For instance in this experiment, where the Car1 desired state does not let the Car2 reach its desired speed. We will now examine the Factored Value MCTS with Max-plus behaviour in this experiment, to examine the improvement in the vehicles' behaviour.

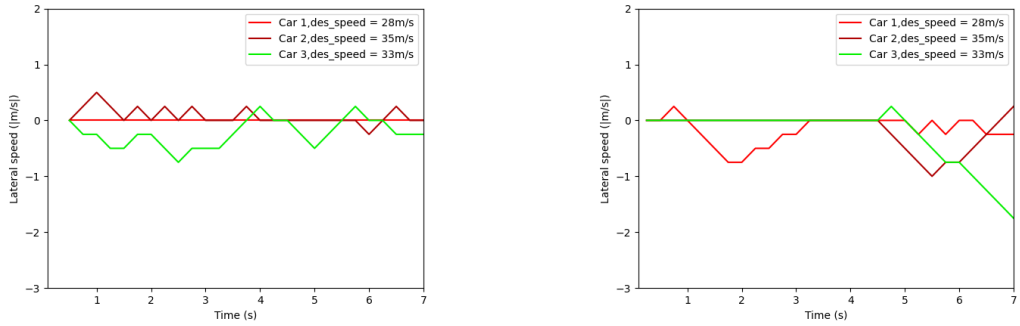
In the scalable tree search algorithm results, Car1 now collaborates with the other two and applies acceleration to the right in order to let Car2 achieve its desired speed. In other words, it anticipates the system's overall goal and takes actions based on that. As we can see in 4.10b, it moves towards the right to help Car1 surpass it. Additionally, after about 3.5 seconds, it decelerates to avoid collision with the Car2, which is moving to the right direction to avoid exceeding the road boundaries. These movements can be seen in more detail at the figures 4.7b and 4.8b.



(A) Plain MCTS case

(B) Scalable Tree Search Case

FIGURE 4.9: Longitudinal speeds of the three vehicles with respect to time



(A) Plain MCTS case (B) Scalable tree search case

FIGURE 4.10: Lateral speeds of the three vehicles with respect to time

The results of this experiment have shown that the FV-MCTS with max-plus algorithm is better than the plain MCTS algorithm in cases where one vehicle has achieved its desired speed and can assist other vehicles with no risk of being involved in a collision, by taking actions like pulling over or accelerate for a small period. In such scenarios, in the plain MCTS implementation the aforementioned vehicle does not take any action to help other vehicles because it has achieved its individual goals and it does not take into consideration other vehicles' desired speeds. In contrast, in the FV-MCTS with max-plus, the vehicle takes more "altrouistic" actions to assist other vehicles achieve their goals, as is evident in this experiment. This "altrouistic" behavior of course needs to coincide with the social welfare of the multiagent system.

Scenario 2

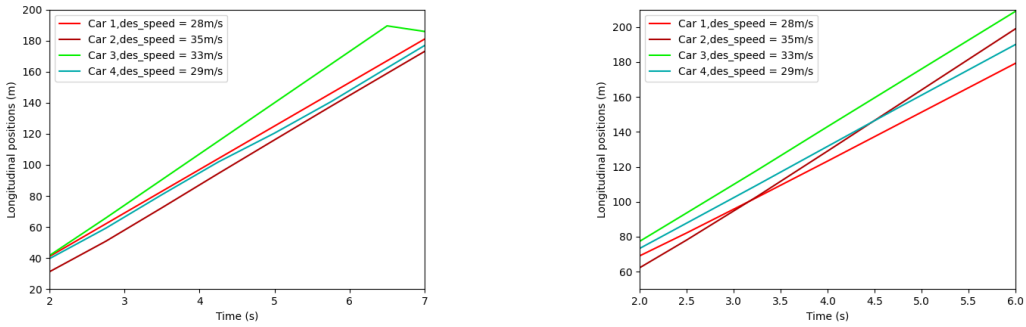
In this scenario, four vehicles are entering the road and are initially placed according to Figure 4.11. The vehicle behind has higher desired speed, so the algorithm has to provide a way of collaboration between the three slower front vehicles blocking the way and the upstream vehicle.

FIGURE 4.11: Initial positions of the three vehicles

In the plain MCTS case, Car4, the middle car of the three downstream cars that are shown in 4.11, initially decelerates in order to avoid collision with vehicles Car1 and Car3, since the other two cars that are very close to it at the right and left. Now, Car2, the vehicle right behind it, also decelerates to avoid crashing with Car4. Figure 4.14a shows the longitudinal speeds of the vehicles, for a specific time horizon. This selfish behaviour of the vehicles results in a large delay before the Car2 can reach a state where it travels with its desired speed and safely, whereas the other

three vehicles achieve their desired speed but are only interested in avoiding collisions between them, a behaviour that makes it more difficult for Car2 to achieve its own desired speed. In the plain MCTS case, the vehicles act selfishly, and in situations like this, they are in an equilibrium state where they are essentially a big obstacle. This completely blocks any faster vehicle upstream that wishes to overtake, such as Car2.

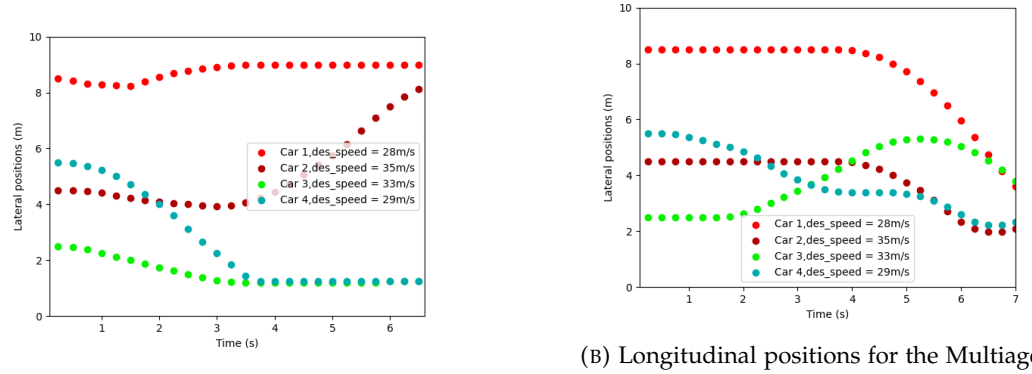
On the contrary, the power of the Factored value MCTS with max-plus algorithm in this scenario can be seen by examining the longitudinal and lateral speeds of the vehicles in Figures 4.14b and 4.15b. In this case, Car3 does not decelerate, because that would lead to a deceleration of the faster vehicle Car2, as happened in the plain MCTS case. In contrast, it drives to the right of the road in order to make space for Car2 to reach its desired speed. Additionally, after roughly the first 200m, as can be seen in Figures 4.12b and 4.13b, the vehicles are aligned at the right of the road, one behind the other, with the two fastest vehicles, Car2 and Car4 proceeding. It is clearly evident in the provided trajectories that the algorithm breaks the blocking formation, so that the faster vehicle behind is accommodated. This behaviour is a desirable one that can only be achieved when applying collaboration between vehicles.



(A) Longitudinal positions for the plain MCTS case

(B) Longitudinal positions for the Multiagent case

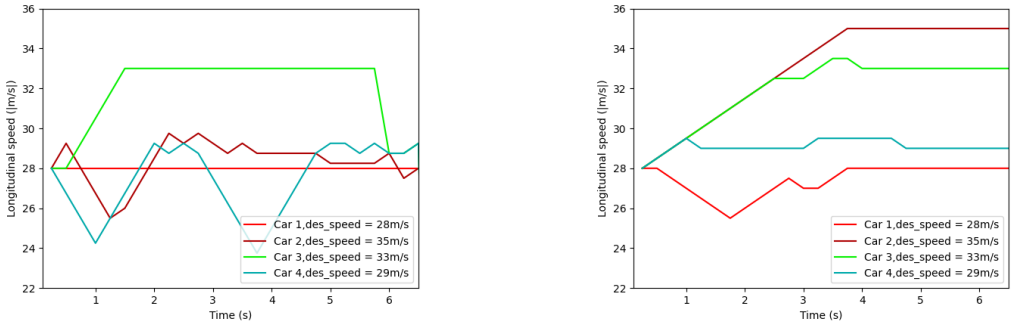
FIGURE 4.12: The longitudinal positions of the four vehicles



(A) Lateral positions for the plain MCTS case

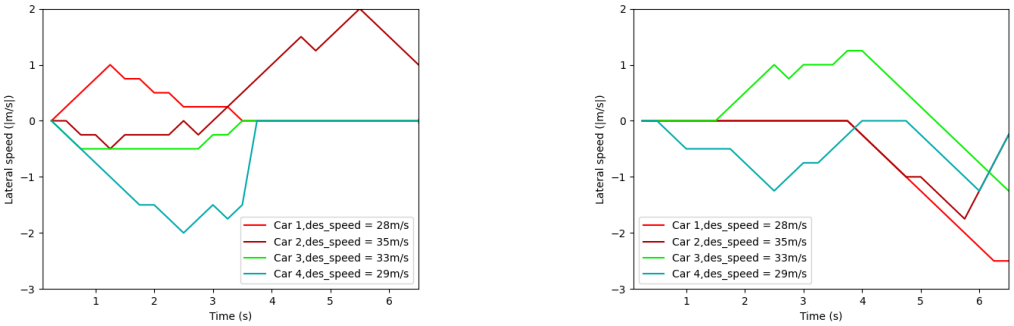
(B) Longitudinal positions for the Multiagent case

FIGURE 4.13: The lateral positions of the four vehicles



(A) Plain MCTS case (B) Scalable Tree Search Case

FIGURE 4.14: Longitudinal speeds of the three vehicles with respect to time



(A) Plain MCTS case (B) Scalable tree search case

FIGURE 4.15: Lateral speeds of the three vehicles with respect to time

It can be easily inferred from this experiment that collision avoidance can be achieved not only when the vehicles are forcing themselves away from the other vehicles, as in the plain MCTS case, but also by communicating their desires and take into consideration the other vehicle's needs and acting accordingly. In fact, in a scenario like the one that is described here, the latter approach has the advantage that results to vehicular speeds closer to the desired ones much faster.

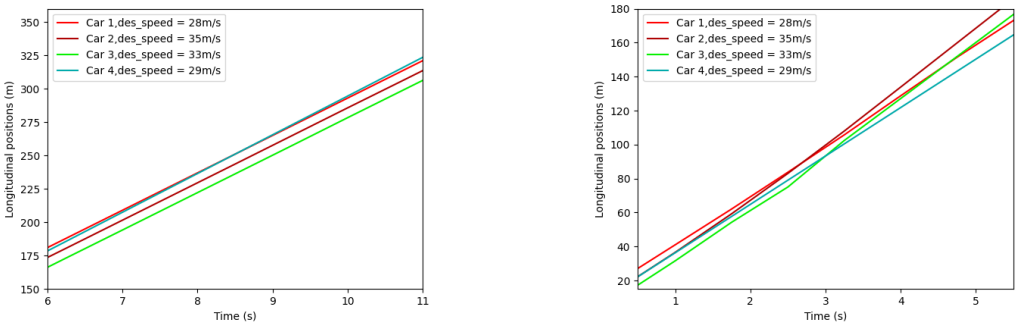
Scenario 3

Another interesting case we examined is when the vehicles are in a state in the road as shown in Figure 4.16.

FIGURE 4.16: Initial positions of the four vehicles

In this scenario the vehicle Car2 is surrounded from the other three vehicles and has the highest desired speed. Car3 is right behind Car2 and has a desired speed smaller than Car2 but higher than the other two cars. Car1 (the vehicle right in front of Car2) and Car4 (the vehicle at the right side of Car2) have already achieved their desired speeds and have no vehicles in front of them, so they do not have profitable moves for themselves, as they only need to retain their speeds.

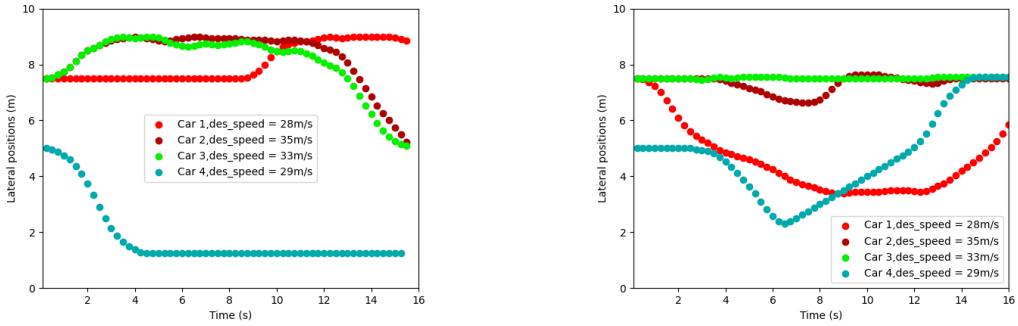
This situation is another case where the plain MCTS algorithm is insufficient to provide a good behaviour, because the vehicles Car1 and Car4 considering only their own profit. As such, they neither accelerate nor move laterally and make space to let Car2 surpass them. Indeed, Figure 4.19a shows that Car2 and Car3 cannot achieve their desired speeds, and instead showcase oscillatory behavior in terms of longitudinal speed. While they attempt to increase speed, they always slow down again due to the slower vehicle Car1 in front. Figure 4.20a shows that although Car3 initially drives to the left of the road to surpass Car1, Car4 in the right does not allow it to make a move to the right, therefore it cannot surpass Car1 considering the road boundary.



(A) MCTS case

(B) Multiagent case

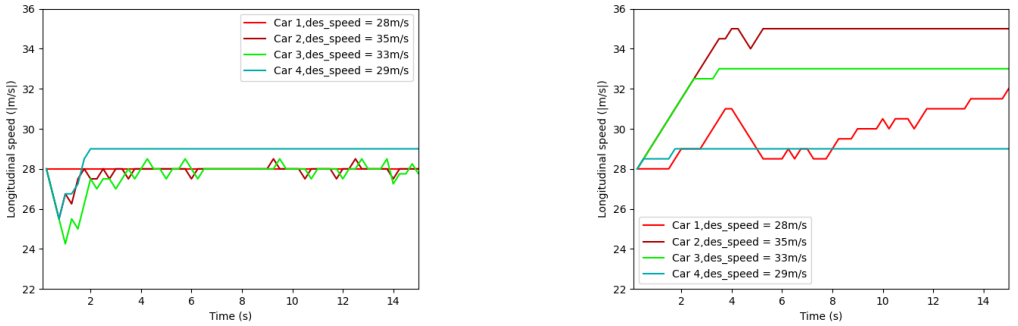
FIGURE 4.17: The longitudinal positions of the four vehicles



(A) MCTS case (B) Multiagent case

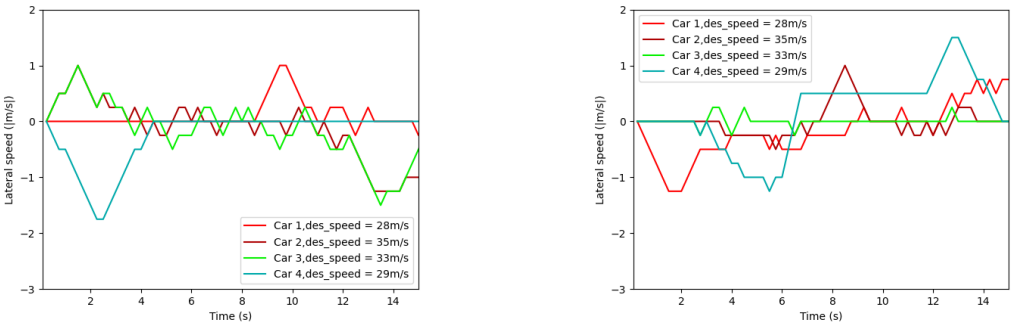
FIGURE 4.18: The lateral positions of the four vehicles

The results of the experiment of the FV-MCTS with max-plus implementation on the other hand, showed that this situation can be tackled appropriately by collaboration between the vehicles. Figure 4.18b shows that Car1 moves to the right in order to let Car2 and Car3 surpass them. In a later stage, in Figure 4.17b it can be seen that after about 7 seconds in the simulation, Car4 goes behind them at the left side of the road, as far away as possible from the vehicle with similar speed, Car1.



(A) Plain MCTS case (B) Scalable Tree Search Case

FIGURE 4.19: Longitudinal speeds of the three vehicles with respect time



(A) Plain MCTS case (B) Scalable tree search case

FIGURE 4.20: Lateral speeds of the three vehicles with respect to time

This is another example, similar to the example of the scenario 2, showing that we have more desireable behavior with the FV-MCTS algorithm for the task of planning in a lane-free vehicular traffic scenario compared to the plain MCTS algorithm. This specific scenario imposes a situation that collaboration between agents is not only desirable, but also critical for achieving satisfying results, something that can be tackled better with FV-MCTS. In this experiment for example, Car1 and Car4 collaborate between them and with the rest of the vehicles, to take actions that benefit the total utility of all vehicles instead of acting selfishly.

4.4 Discussion

The single-agent approach has very good results in scenarios where the vehicle capacity of the traffic scenario exceeds the capacity of lane-based scenarios, with a noticeable though small impact in traffic safety. The effectiveness of the algorithm increases when more simulations are run and bigger trees are created, with a decline in performance in traffic capacities larger than the largest traffic capacity of comparable lane-based traffic scenarios.

One key characteristic of the MCTS, that according to the results of the experiments is suitable for lane-free traffic settings, is that it can perform very well in many different scenarios and can adapt its behaviour to different vehicular traffic capacities. The vehicles are faster in traffic settings with low capacities and more careful in traffic setting with higher capacities, a behaviour that the algorithm exhibits by itself and has not been modelled in advance. The saturation in performance comes when the traffic capacity is so high that there are no suitable actions that would avoid collisions and achieve the desired speed at the same time. The saturation point, the capacity above which the performance of the algorithm is getting worse, in terms of average delay and average deviation from the desired speed, can be improved by inserting a wider range of discrete available actions than the 21 that we have in the experiments. However, this would lead to a larger decision time for every vehicle.

The multi-agent scenarios, showed that the FV-MCTS algorithm potential collision avoidance is great and can provide better results from the single-agent approach in specific difficult scenarios. The agents in this formulation, avoid collisions and obstacles in a smart way, in cases where the single-agent approach cannot avoid them, not only by forcing themselves away from each other, but also by communicating their intentions and taking other vehicles' intentions into consideration. In the multi-agent algorithm where the vehicles communicate their intentions, the vehicles don't blindly take actions by estimating other vehicles positions and velocities, but also consider the impact of these actions in the relations between them and the other vehicles. If a considered action of the vehicle could worsen other vehicles' state, either by increasing their probability of collision or by making them not achieving their desire speeds, this action is considered as less desired. This behaviour is making the FV-MCTS with maxplus algorithm worth further investigation in the domain, since the performance of the multiagent variant in very high capacity scenarios was not examined in this thesis.

Chapter 5

Conclusions and future work

Two existing algorithms where adapted for trajectory planning in a lane-free traffic environment. For the first algorithm, the well known Monte Carlo Tree Search (MCTS) algorithm, the experimental results were very promising and showed a great potential for probabilistic planning algorithms in the domain, both in terms of safety and desired travelling.

For the second algorithm, the collaborative multiagent nature of the lane-free vehicular traffic environment was taken into consideration, together with the anytime-time planning of the MCTS. The experiments showed that for certain situations where the MCTS algorithm cannot deliver satisfying results, a Multiagent approach based on it can provide better solutions. Although the experimental results were promising, the experimental evaluation of this approach was not thorough enough and more investigation must be done in future work towards improving the computational effort of the approach.

5.1 Future Work

The lane-free traffic setting is a novel paradigm, therefore a detailed comparison of the existing works and this work could provide some very interesting conclusions about the nature of the domain and the possible directions of future work, regarding it. For example, the effectiveness of other Monte Carlo based methods could be investigated together with this work and give a measure of the impact of these methods in the domain.

In the context of trajectory planning, there are several sophisticated planning methods or MCTS based alternatives that could be adapted for trajectory planning in lane-free traffic environments. A combination of MCTS and machine learning could provide trajectory prediction of other vehicles, like the one proposed in the game Hex [17]. Moreover, a hierarchical Monte Carlo tree search approach [47] could provide a way to effectively search in very large state spaces which would allow us to adapt a bigger actions space.

In the context of MMDP environments in general, many interesting ideas could be adapted for the domain. Specifically, our work in MMDP requires a predefined coordination graph to be solved. An interesting extension would be an approach that dynamically learns the coordination graph via interaction with the lane-free model [28]. This approach would potentially provide a framework suitable for a wide range of lane-free scenarios, like complex routes and extremely dense traffics, cases when our approach is insufficient. However, learning the coordination graph itself would pose a bottleneck for large-scale simulations, but we believe it is a worthwhile investigation, especially in scenarios as the ones examined in this work.

Other interesting MMDP approaches with decomposition of the global reward function $Q(s, a)$ could be adapted for the problem. An interesting direction in the aforementioned framework would be the investigation of algorithms involving learning the reward function, like learning deep coordination graphs [5] or sparse cooperative Q-Learning [39] for the planning problem in hand and how they tackle more complex lane-free traffic situations. Finally, a more comprehensive study of the impact of the exploration terms to the max-plus convergence could help us better understand the complexity of a multiagent lane-free traffic environment [10].

Bibliography

[1] "Agent Organization Framework". In: *Organizational Principles for Multi-Agent Architectures*. Basel: Birkhäuser Basel, 2005, pp. 9–41. ISBN: 978-3-7643-7318-4. DOI: 10.1007/3-7643-7318-0_2. URL: https://doi.org/10.1007/3-7643-7318-0_2.

[2] Christopher Amato and Frans A Oliehoek. "Scalable planning and learning for multiagent POMDPs: Extended version". In: *arXiv preprint arXiv:1404.1140* (2014).

[3] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[4] Henrik Bey, Frank Dierkes, Sebastian Bayerl, Alexander Lange, Dennis Faßender, and Jörn Thielecke. "Optimization-based tactical behavior planning for autonomous freeway driving in favor of the traffic flow". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1033–1040.

[5] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. "Deep coordination graphs". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 980–991.

[6] Craig Boutilier. "Planning, Learning and Coordination in Multiagent Decision Processes". In: *Theoretical Aspects of Rationality and Knowledge*. 1996.

[7] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4:1 (Mar. 2012), pp. 1–43. DOI: 10.1109/TCIAIG.2012.2186810.

[8] R. Parr C. Guestrin D. Koller and S. Venkataraman. "Efficient Solution Algorithms for Factored MDPs." In: Journal of Artificial Intelligence Research 19 (Oct. 2003), 399–468. (2003). DOI: https://doi.org/10.1613/jair.1000.

[9] Jienan Chen, Cong Zhang, Jinting Luo, Junfei Xie, and Yan Wan. "Driving Maneuvers Prediction Based Autonomous Driving Control by Deep Monte Carlo Tree Search". In: *IEEE Transactions on Vehicular Technology* 69.7 (2020), pp. 7146–7158. DOI: 10.1109/TVT.2020.2991584.

[10] Shushman Choudhury, Jayesh K Gupta, Peter Morales, and Mykel J Kochenderfer. "Scalable anytime planning for multi-agent mdps". In: *arXiv preprint arXiv:2101.04788* (2021).

[11] W.W. Hager C.L. Dardy and A.V. Rao. "An hp-adaptive pseudospectral method for solving optimal control problems". In: *Optimal control application and methods* 32 (2011), pp. 476–502.

[12] Carlos Guestrin Daphne Koller Ronald Parr. "Multiagent planning with factored MDPs". In: Advances in Neural information processing systems-MIT Press (2002).

[13] Institute of Disability UCED. "Annual report on people with disabilities in America". In: (Jan. 2019). URL: https://disabilitycompendium.org/sites/default/files/user-uploads/Annual_Report_2018_Accessible_AdobeReaderFriendly.pdf.

[14] Chiyu Dong, John M Dolan, and Bakhtiar Litkouhi. "Intention estimation for ramp merging control in autonomous driving". In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 1584–1589.

[15] J.lu E.Velenis P.Tsiotras. "Aggressive maneuvers on loose surfaces;Data analysis and input parametrization". In: *Mediterranean conference on control automation* (2007), pp. 1–6.

[16] Sara Fleury, Philippe Soueres, J-P Laumond, and Raja Chatila. "Primitives for smoothing mobile robot trajectories". In: *IEEE transactions on robotics and automation* 11.3 (1995), pp. 441–448.

[17] Chao Gao, Ryan Hayward, and Martin Müller. "Move prediction using deep convolutional neural networks in Hex". In: *IEEE Transactions on Games* 10.4 (2017), pp. 336–343.

[18] Dimitrios Troullinos Georgios Chalkiadakis Ioannis Papamichail and Markos Papageorgiou. "Collaborative Multiagent Decision Making for Lane- Free Autonomous Driving". In: *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. May 2021.

[19] Timothy Ha, Kyunghoon Cho, Geonho Cha, Kyungjae Lee, and Songhwai Oh. "Vehicle Control with Prediction Model Based Monte-Carlo Tree Search". In: *17th International Conference on Ubiquitous Robots (UR)*. 2020, pp. 303–308. DOI: 10.1109/UR49135.2020.9144958.

[20] David Hsu, J-C Latombe, and Rajeev Motwani. "Path planning in expansive configuration spaces". In: *Proceedings of international conference on robotics and automation*. Vol. 3. IEEE. 1997, pp. 2719–2726.

[21] Tommi Jaakkola, Satinder Singh, and Michael Jordan. "Reinforcement learning algorithm for partially observable Markov decision problems". In: *Advances in neural information processing systems* 7 (1994).

[22] D.Kirkpatrick J.Backer. "Finding curvature constrained paths that avoid polygonal obstacles". In: *Proceedings of the 23rd annual symposium on computational geometry* (2007), pp. 66–73.

[23] M.Sharir J.T.Schwartz. "On the piano movers problem;general techniques for computing topological properties of real algebraic manifolds". In: *Advances in applied mathematics* 4 (1983), pp. 298–351.

[24] Minoru KAMATA. "Current Status and Future Outlook of Automated Driving". In: *ClassNK technical journal* 2021.2 (2021), pp. 1–9.

[25] Athanasia Karalakou, Dimitrios Troullinos, Georgios Chalkiadakis, and Markos Papageorgiou. "Deep RL Reward Function Design for Lane-Free Autonomous Driving". In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection*. Ed. by Frank Dignum, Philippe Mathieu, Juan Manuel Corchado, and Fernando De La Prieta. Cham: Springer International Publishing, 2022, pp. 254–266. ISBN: 978-3-031-18192-4.

[26] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Machine Learning: ECML 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293. ISBN: 978-3-540-46056-5.

[27] Lanxin Lei, Ruiming Luo, Renjie Zheng, Jingke Wang, JianWei Zhang, Cong Qiu, Liulong Ma, Liyang Jin, Ping Zhang, and Junbo Chen. "KB-Tree: Learnable and Continuous Monte-Carlo Tree Search for Autonomous Driving Planning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE Press, 2021, 4493–4500. DOI: 10.1109/IROS51168.2021.9636442. URL: https://doi.org/10.1109/IROS51168.2021.9636442.

[28] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. "Deep implicit coordination graphs for multi-agent reinforcement learning". In: *arXiv preprint arXiv:2006.11438* (2020).

[29] Qi Liu, Xueyuan Li, Shihua Yuan, and Zirui Li. "Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 30–37.

[30] Milad Malekzadeh, Diamantis Manolis, Ioannis Papamichail, and Markos Papageorgiou. "Empirical Investigation of Properties of Lane-free Automated Vehicle Traffic". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022, pp. 2393–2400. DOI: 10.1109/ITSC55140.2022.9921864.

[31] Hootan Nakhost and Martin Müller. "Monte-Carlo Exploration for Deterministic Planning." In: *IJCAI*. Vol. 9. 2009, pp. 1766–1771.

[32] NHTSA. "Critical reasons for crashes investigated in the national motor vehicle crash causation survey". In: *Traffic Safety Facts* (Feb. 2015). URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115.

[33] J.R.Kok N.Vlassis R.Elhorst. "Anytime Algorithms for Multiagent Decision Making using Coordination Graphs". In: IEEE International Conference on Systems, Man and Cybernetics 1 (2004), pp. 953–957.

[34] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55. DOI: 10.1109/TIV.2016.2578706.

[35] Markos Papageorgiou, Kyriakos-Simon Mountakis, Iasson Karafyllis, Ioannis Papamichail, and Yibing Wang. "Lane-Free Artificial-Fluid Concept for Vehicular Traffic". In: *Proceedings of the IEEE* 109.2 (2021), pp. 114–121. DOI: 10.1109/JPROC.2020.3042681.

[36] Dimitri P.Bertsekas. "Dynamic programming and optimal control". In: *Athena Scientific Belmont* 1 (2005).

[37] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". In: International Conference on Machine Learning (ICML), 4295-430 (2018).

[38] R.Bishop. "A survey of intelligent vehicle applications worldwide". In: *Proceedings of the IEEE Intelligent Vehicles Symposium* (2000), pp. 25–30. URL: https://www.semanticscholar.org/paper/A-survey-of-intelligent-vehicle-applications-Bishop/b80f4f07fa0503fc980a746c7d8c05d9abf9e52c.

[39] Jelle R.Kok and Nikos Vlassis. "Sparse cooperative Q-Learning". In: 21st international conference of Machine Learning, Banff,Canada, July 2004.

[40] M. Schoenauer et.All S. Gelly L. Kocsis. "The grand challenge of computer go: Monte carlo tree search and extensions". In: *Commun.ACM* 55 (Mar. 2012), pp. 106–113. URL: https://hal.inria.fr/hal-00695370v3/document.

[41] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[42] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, 2085–2087.

[43] Zineb Tahiri, K. Jetto, Marouane Bouadi, Abdelilah Benyoussef, and Abdallah Kenz. "The effect of anisotropy on the traffic flow behavior: Investigation of the correlation created by a single node on two-lane roads". In: *International Journal of Modern Physics C* 31 (Jan. 2020). DOI: 10.1142/S0129183120500606.

[44] Justin K Terry, Nathaniel Grammel, Sanghyun Son, and Benjamin Black. "Parameter sharing for heterogeneous agents in multi-agent reinforcement learning". In: *arXiv e-prints* (2020), arXiv–2005.

[45] Dimitrios Troullinos, Georgios Chalkiadakis, Diamantis Manolis, Ioannis Papamichail, and Markos Papageorgiou. "Lane- Free Microscopic Simulation for Connected and Automated Vehicles". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3292–3299. DOI: 10.1109/ITSC48978.2021.9564637.

[46] S.M.La Valle. "Rapidly-exploring random trees a new tool for path planning". In: *Technical Presentation CS Department Iowa State University* (1998).

[47] Ngo Ahn Vien and Marc Toussaint. "Hierarchical Monte-Carlo planning". In: *29th AAAI conference on Artificial Intelligence* 29 (2015).

[48] Nicola Catenacci Volpi, Yan Wu, and Dimitri Ognibene. "Towards event-based MCTS for autonomous cars". In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2017, pp. 420–427.

[49] Venkata Karteek Yanumula, Panagiotis Typaldos, Dimitrios Troullinos, Milad Malekzadeh, Ioannis Papamichail, and Markos Papageorgiou. "Optimal Path Planning for Connected and Automated Vehicles in Lane-free Traffic". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3545–3552. DOI: 10.1109/ITSC48978.2021.9564698.

[50] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G Keller, et al. "Making bertha drive—an autonomous journey on a historic route". In: *IEEE Intelligent transportation systems magazine* 6.2 (2014), pp. 8–20.