Cooperative Planning and Control for Connected and Automated Vehicles' On-ramp Merging in Mixed Traffic Through Value Decomposition-based Multiagent Deep Reinforcement Learning

Yuteng Zhang

Transport, Infrastructure, Logistics
Master thesis

Cooperative Planning and Control for Connected and Automated Vehicles' On-ramp Merging in Mixed Traffic Through Value Decomposition-based Multiagent Deep Reinforcement Learning

by

Yuteng Zhang

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 22 November.

Student number:	5721903	
Project duration:	April 3, 2024 – November 22, 2024	
Thesis committee:	Dr. ir. H. Farah,	Chair, Civil Engineering and Geosciences, TU Delft
	Dr. ir. M. Rinaldi,	Supervisor, Civil Engineering and Geosciences, TU Delft
	Dr. ir. B. Shyrokau,	Supervisor, Mechanical Engineering, TU Delft
	Dr. ir. L. Ferranti,	External Supervisor, Mechanical Engineering, TU Delft
	Y. Dong,	Daily Supervisor, Civil Engineering and Geosciences, TU Delft
	C. Evans,	Daily Supervisor, Civil Engineering and Geosciences, TU Delft

Preface

When I write down these words, I realize that I'm actually approaching the end of my study life at TU Delft. Looking back on these two years of study, it has been a journey filled with various emotions - joy, bitterness, and challenges. Throughout this journey, I am both proud and excited that I have ultimately stuck with it and persevered through to the end.

I would like to express my deepest gratitude to my committee for their guidance over the past seven months. I am particularly thankful to my daily supervisors, Yongqi and Callum, who have been actively involved since the initial conceptualization of this project, offering professional insights throughout. Their consistent weekly meetings provided me with the motivation and supervision needed to stick with the project - in my view, the most crucial factor for success, as the journey itself matters more than reaching the destination. I am especially grateful to Marco for his incisive suggestions, as his ability to provide enlightening directions at critical moments has been invaluable, from designing the reward function to increasing action frequency and simplifying the state-action space. As a naive engineer and researcher, I have gained tremendous insights from his guidance. My sincere appreciation goes to my chairperson, Haneen, for accepting to supervise me and offering the opportunity to work on this innovative project; her efforts in assembling such a well-rounded committee and her meticulous arrangement of milestone meetings have been crucial to the successful completion of this project. To Barys, I extend my heartfelt thanks for his suggestions from a vehicle engineering perspective, particularly his suggestion for detailed analysis of low-level control in the evaluation section, which has made my report more comprehensive, as the actual performance on the road is what matters most.

I would also like to express my heartfelt gratitude to my parents for their unwavering support, both emotionally and financially, in pursuing my studies in the Netherlands. Their love and support remained constant, showing understanding and encouragement even during my extended study period. My sincere thanks also go to my friends, who have enriched my master's life in countless ways.

Yuteng Zhang
Delft, November 2024

Executive Summary

Introduction

The rapid advancement of autonomous vehicle (AV) technology is poised to revolutionize transportation systems in the coming decades. With 20-40% of vehicles expected to be automated by 2030, a critical challenge emerges in managing mixed traffic environments where autonomous and human-driven vehicles (HDVs) coexist. This is particularly evident in on-ramp merging scenarios, where efficient coordination between vehicles is crucial for maintaining traffic flow and safety.

Traditional approaches to this challenge fall into two categories: rule-based methods and optimization-control strategies. Rule-based methods, while effective in controlled environments, struggle with the variability of real traffic conditions. Optimization-based control strategies, such as Model Predictive Control (MPC), require precise modeling of vehicle dynamics and substantial computational resources. More recent approaches such as learning-based methods, particularly Multi-Agent Reinforcement Learning (MARL), have shown promise in handling these complex scenarios but face challenges in credit assignment - accurately determining each agent's contribution to overall system performance.

A new method has been developed for cooperative on-ramp merging to improve credit assignment, and safety performance while maintaining traffic efficiency.

Method

Our research methodology employs a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework to model the multi-agent on-ramp merging scenario. This approach recognizes that each agent (vehicle) has limited information about its environment and must make decisions based on partial observations.

The QMIX algorithm, which forms the foundation of my method, addresses the credit assignment challenge through value function decomposition - breaking down the overall system value into individual agent contributions while maintaining certain constraints. I enhance this base algorithm with two significant improvements, shown in Figure 1. The first is the Q(λ) Return, a mechanism that balances immediate and long-term rewards by combining Temporal Difference (TD) learning with Monte Carlo methods, helping in more efficient value estimation and improved learning stability. The second is the Action Mask, a safety-oriented mechanism that prevents agents from selecting potentially dangerous actions by evaluating their consequences before execution.

The reward function integrates multiple objectives with different weights. The primary objectives focus on collision avoidance and maintaining safe distances between vehicles for safety. Secondary objectives include optimizing speed, minimizing unnecessary lane changes, and encouraging timely merging behaviors, all contributing to smooth traffic flow.

Results and Conclusion

The evaluation compared QMIX-QLambdaM against three state-of-the-art baselines: QMIX, Multi-Agent Advantage Actor-Critic (MAA2C), and Counterfactual Multi-Agent (COMA, a counterfactual baseline policy gradient method). The comparison was conducted across different traffic densities.

In low-density traffic (9.09 veh/km), QMIX-QLambdaM achieved a 100% reduction in collision rate compared to all baselines, while improving average speed by 10.8% over QMIX, 16.0% over MAA2C, and 80.3% over COMA.

In medium-density scenarios (13.64 veh/km), the safety performance maintained its 100% collision reduction, with efficiency improvements of 7.6% over QMIX, 7.1% over MAA2C, and 58.2% over COMA in terms of average speed.

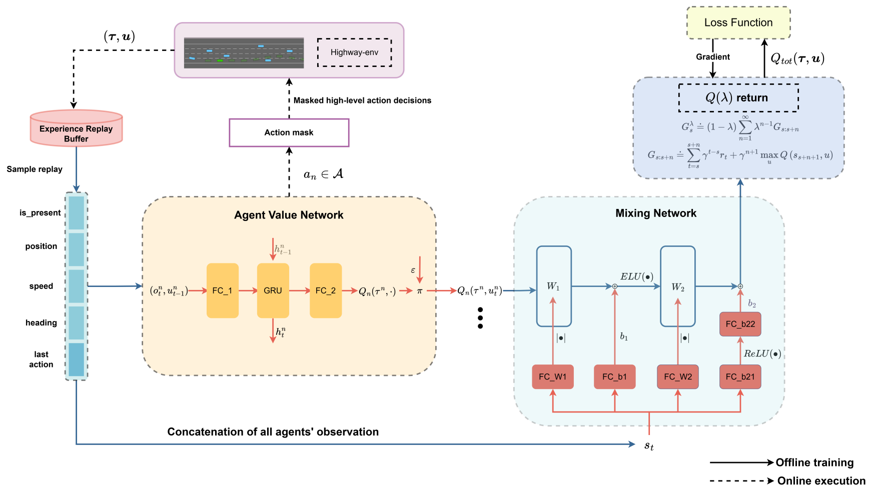


Figure 1: QMIX-QLambdaM

For high-density traffic (18.18 veh/km), the proposed method continued to demonstrate robust performance, maintaining the 100% collision reduction while achieving speed improvements of 11.3% over QMIX, 11.9% over MAA2C, and 54.5% over COMA. In terms of total reward, QMIX-QLambdaM demonstrated improvements of 91 points over QMIX, 104 points over MAA2C, and 123 points over COMA.

The learning efficiency of the improved algorithm showed marked improvements, with a 53% increase in mean total reward compared to the original QMIX implementation. The learning curves demonstrated superior credit assignment capabilities, evidenced by consistently higher and more stable reward curves throughout the training process, particularly after 7000 episodes. The action masking mechanism proved particularly effective, reducing initial collision rates by 70.9% compared to baseline methods.

These results demonstrate that QMIX-QLambdaM successfully addresses the credit assignment problem in cooperative on-ramp merging while maintaining superior safety and efficiency. The consistent performance across varying traffic densities, particularly the achievement of zero collisions and significant speed improvements, indicates that the proposed enhancements effectively balance safety and efficiency objectives. The marked improvement in learning efficiency further suggests that the combination of Q(λ) return and action masking mechanisms significantly enhances the algorithm's practicality for real-world applications. These findings establish QMIX-QLambdaM as a promising approach for managing autonomous vehicle merging in mixed traffic environments.

Recommendations

Future scientific research could advance QMIX-QLambdaM in several key directions. The first priority is comprehensive hyperparameter fine-tuning, particularly focusing on the RNN network dimensions, mixing network structure, and replay buffer size, which have shown a significant impact on performance in previous studies. Transfer learning where there is the replacement of RNN architecture with transformer models presents another promising direction, to make the policy in handling variable numbers of agents through their attention mechanism more realistic. Moreover, future research should expand to more diverse and realistic traffic scenarios, particularly examining how agents with different reward structures - from purely self-interested to cooperative behaviors - affect overall traffic dynamics. Additionally, the integration of more sophisticated human driver modeling is essential, focusing on capturing the uncertainty and dynamic nature of human driving patterns through probabilistic behavior models.

For road infrastructure implementation, several specific technologies are recommended. Road-Side Units (RSUs) should be installed at critical merging points, equipped with high-resolution cameras and LiDAR sensors for real-time traffic monitoring. These units would need to be integrated with edge computing devices capable of processing sensor data and communicating with vehicles through Cellular Vehicle-to-Everything (C-V2X) protocols.

Vehicle manufacturers need to focus on specific hardware and software implementations. The integration of QMIX-QLambdaM requires on-board computing units with sufficient processing power, such as NVIDIA DRIVE AGX or Intel Mobileye platforms. Vehicle-to-Everything (V2X) communication modules should be standardized, incorporating both DSRC and C-V2X capabilities to ensure broad compatibility. Low-level control systems should be enhanced with sophisticated controllers and adaptive gain scheduling to ensure smooth trajectory following and a comfortable passenger experience.

Contents

List of Figures

List of Tables

List of Abbreviations

AV	Autonomous Vehicle
CAV	Connected and Automated Vehicle
COMA	Counterfactual Multi-Agent
CTDE	Centralized Training with Decentralized Execution
Dec-POMDP	Decentralized Partially Observable Markov Decision Process
DDQN	Deep Double Q-Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DTDE	Decentralized Training with Decentralized Execution
GRU	Gated Recurrent Unit
HDV	Human-Driven Vehicle
IDM	Intelligent Driver Model
IGM	Individual-Global-Max
LSTM	Long Short-Term Memory
MAA2C	Multi-Agent Advantage Actor-Critic
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MARL	Multi-Agent Reinforcement Learning
MC	Monte Carlo
MDP	Markov Decision Process
MPC	Model Predictive Control
NGSIM	Next Generation Simulation
PID	Proportional-Integral-Derivative
RNN	Recurrent Neural Network
SARSA	State-Action-Reward-State-Action
SMAC	StarCraft Multi-Agent Challenge
TD	Temporal Difference
TTC	Time To Collision
V2C	Vehicle-to-Cloud
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
V2X	Vehicle-to-Everything
VDN	Value Decomposition Network

1 Introduction

1.1 Background

Autonomous vehicles (AVs) stand at the forefront of a transportation revolution, poised to bring about transformative changes across multiple facets of mobility. Research indicates that the widespread implementation of AV technology has the potential to fundamentally reshape the landscape of road travel (Deichmann 2023). This paradigm shift promises to enhance not only safety standards but also to significantly boost efficiency and elevate the overall user experience. The advent of AVs is anticipated to redefine the concept of personal transportation, offering the prospect of more streamlined and enjoyable journeys for a diverse user base. Beyond individual travel, the integration of AVs into public transit systems presents a compelling solution to longstanding challenges of accessibility and environmental sustainability. This integration holds particular promise for demographics that have historically encountered barriers in accessing conventional transportation options, thereby fostering a more inclusive and equitable mobility ecosystem (Irshayyid, J. Chen, and Xiong 2024). The ramifications of AV technology extend far beyond personal convenience, potentially catalyzing transformative changes in urban planning strategies, mitigating traffic congestion, and contributing to the development of more sustainable urban environments. As AV technology continues to evolve and mature, it not only promises to revolutionize our modes of travel but also to fundamentally alter our perception and interaction with urban spaces, heralding a new era of smart, efficient, and inclusive urban mobility.

Among the various applications of autonomous vehicles (AVs), the on-ramp merging scenario has garnered increasing attention due to its critical role in traffic management. This scenario is particularly significant as it often creates bottlenecks, increases accident risks, and compromises the overall safety and efficiency of transportation networks. The complexity of on-ramp merging lies in its demand for precise coordination between multiple vehicles. Vehicles in the main lane near the merging point must dynamically adjust their speeds to create sufficient space for incoming vehicles. Simultaneously, vehicles on the on-ramp face the challenge of calibrating their speed and executing timely lane changes to seamlessly integrate into the main traffic flow, all while avoiding potential deadlocks (Irshayyid, J. Chen, and Xiong 2024). These intricate interactions underscore the crucial need for sophisticated coordination and cooperation mechanisms in controlling both the main lane and merging vehicles. To address these challenges, state-of-the-art approaches are increasingly turning to connected autonomous vehicles (CAVs). By leveraging cutting-edge communication technologies such as vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-cloud (V2C) communications (Dey et al. 2016), CAVs significantly enhance collective decision-making capabilities, thereby improving the efficiency and safety of the merging process.

Traditional approaches to this problem can be categorized as rule-based methods and optimization-control strategies. For the rule-based methods relying on predefined, hard-coded rules, (Min et al. 2021) noted that while these methods may work in controlled environments, they struggle with the variability of real traffic conditions, such as unexpected congestion or accidents, which require more dynamic and responsive control strategies. In contrast, optimization-based control strategies, such as Model Predictive Control (MPC), employ detailed dynamic models to represent vehicle interactions. Research by (Irshayyid, J. Chen, and Xiong 2024) suggests that while these methods show promise, they have significant drawbacks. They rely heavily on precise modeling, even for unpredictable human-driven vehicles, and often require substantial computational resources due to solving complex optimization problems at each time step. In addressing the complex challenges of on-ramp merging, cutting-edge research is increasingly turning to Deep Reinforcement Learning (DRL), an advanced extension of traditional reinforcement learning techniques (Irshayyid, J. Chen, and Xiong 2024) with the neural network serving as a powerful function approximator. The appeal of DRL in this context lies in its data-driven nature, which allows for the continuous refinement of control policies through direct interaction with the environment, avoiding the need for precise system modeling. This characteristic is particularly advantageous in the dynamic and unpredictable realm of traffic management. However, the downside of pure RL is it only considers one agent which can not capture the intrinsic demand of coordination between main-lane traffic and merging traffic. That is how the MARL contributes to the realm. MARL offers distinct advantages over pure RL in addressing on-ramp merging. It naturally models multi-vehicle interactions, enables cooperative behavior, handles partial observability more effectively, and provides scalable solutions for complex traffic systems. These features make MARL particularly suited for the collaborative nature of efficient and safe merging strategies.

Referred to (Zong 2019), while the potential of fully autonomous transportation systems is compelling, 20-40% of vehicles are expected to be automated by 2030 and the full penetration could be realistic in a few decades, therefore the current reality presents a more nuanced challenge: the mixed traffic environment. This transitional phase, where CAVs coexist with human-driven vehicles (HDVs) on both mainline and ramp roadways, introduces a new layer of complexity to traffic management, particularly in on-ramp merging scenarios, presenting unique challenges for on-ramp merging. The unpredictability of human drivers and the communication gap between autonomous and human-driven vehicles add significant complexity. This heterogeneous setting demands more sophisticated control strategies that can adapt to varied behaviors while ensuring safety and efficiency for all road users.

1.2 Literature Review

The development of effective strategies for CAV on-ramp merging has been a focus of research for decades, evolving from classical rule-based and optimization methods to more recent learning-based approaches. This section provides an overview of these methods, highlighting their strengths and limitations, and ultimately demonstrating the need for advanced techniques like MARL.

1.2.1 Classical Methods for CAVs On-ramp Merging

Classical approaches to the CAV on-ramp merging problem have laid the foundation for current research, offering valuable insights into the complexities of traffic management. These methods can be broadly categorized into two primary types (Rios-Torres and Malikopoulos 2017): heuristic rule-based approaches and optimization-control strategies, each with its own set of advantages and challenges.

1.2.1.1 Rule-based Methods

Rule-based methods rely on predefined heuristics derived from the study of nonlinear system dynamics. The evolution of these methods reflects a progression from simple hierarchical structures to more sophisticated, communication-based systems.

(Schmidt and Posch 1983) pioneered this approach with a two-layer hierarchical control strategy, assigning merging order based on projected entry times and calculating necessary accelerations. A significant advancement came with (Uno, Sakaguchi, and Tsugawa 1999), who introduced the concept of virtual vehicles, mapping a virtual vehicle onto the main road before the actual merging point. As vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications evolved, (Marinescu et al. 2012) expanded on slot-based traffic management, incorporating these capabilities to improve throughput and reduce delays. In contrast, (Antoniotti, Desphande, and Girault 1997) proposed a decentralized hybrid controller where vehicles make merging decisions based solely on local sensor information, addressing scenarios with limited inter-vehicle communication.

(Min et al. 2021) noted that while these methods may work in controlled environments, they struggle with the variability of real traffic conditions, such as unexpected congestion or accidents, which require more dynamic and responsive control strategies.

1.2.1.2 Optimization-control Methods

Optimization-control methods rely on accurate mathematical models of vehicle dynamics, formulating and resolving model-based optimization problems. The field has progressed from early linear models to complex, multi-objective optimization frameworks.

(Athans 1969) laid the groundwork by formulating the merging problem as a linear optimal regulator. As computational capabilities improved, more complex methods emerged. (Duret, M. Wang, and Ladino 2020) introduced a layered control strategy with a top-level controller for vehicle sequencing and a subordinate controller for acceleration optimization. The advent of Model Predictive Control (MPC) is exemplified by (Cao et al. 2015), who proposed a multi-layer optimization framework for cooperative decision-making in mixed traffic systems. Most recently, (Sun, Huang, and Zhang 2020) developed a dual-layer optimization framework, further refining trajectory planning and merging order optimization in mixed traffic environments.

(Irshayyid, J. Chen, and Xiong 2024) suggest that while these methods show promise, they have significant drawbacks. They rely heavily on precise modeling, even for unpredictable HDVs, and often require substantial computational resources due to solving complex optimization problems at each time step.

1.2.2 Learning-based Methods for CAVs On-ramp Merging

The limitations of classical methods have led researchers to explore more adaptive and data-driven approaches, particularly in light of recent advancements in computational capabilities and data collection techniques. DRL has emerged as a promising solution, offering robust performance in complex, dynamic environments such as on-ramp merging scenarios.

DRL's strength lies in its ability to learn optimal decision-making policies through continuous interaction with the environment, as illustrated in Figure 1.1. This approach allows the agent to adapt to changing conditions and unpredictable behaviors, addressing many of the challenges faced by traditional methods. In the context of on-ramp merging, DRL has demonstrated excellent performance, with research progressing from single-agent to multi-agent approaches to better reflect the complexity of real-world traffic situations.

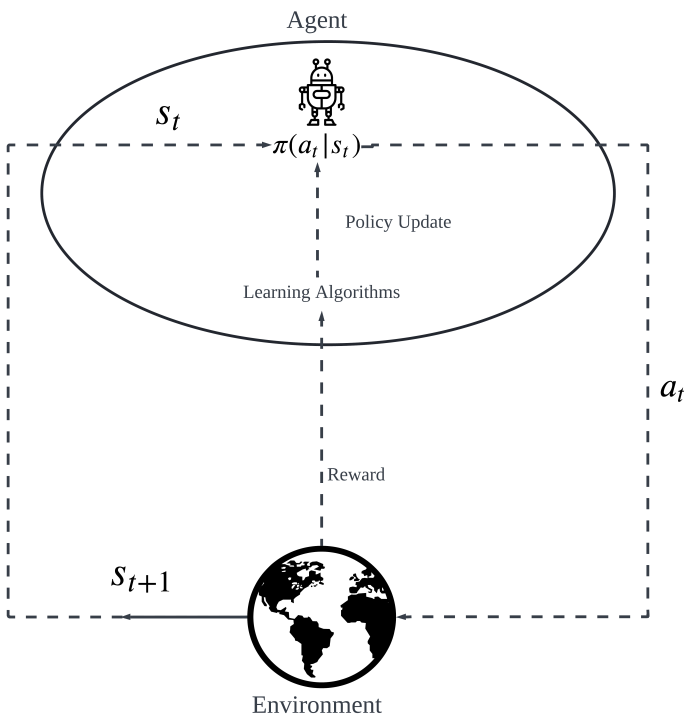


Figure 1.1: Illustration of RL

1.2.2.1 Single-Agent Reinforcement Learning

Early AV research primarily concentrated on single-agent reinforcement learning frameworks, where researchers studied interactions between one AV and HDVs. This simplified approach helped scientists better understand individual autonomous agent behavior in isolation. For instance, (Triest, Villaflor, and Dolan 2020) utilized the NGSIM dataset to analyze the behavior of a single autonomous vehicle across hundreds of highway merging scenarios, with other vehicles following fixed historical trajectories rather than dynamically responding to the AV. (P. Wang and Chan 2017) combined deep learning architectures like LSTM with reinforcement learning methods such as DQN to develop effective merging strategies while leveraging long-term historical data. (Bouton et al. 2019) explored how autonomous vehicles could handle uncertainty when merging, particularly regarding the unpredictable behavior of surrounding vehicles. Their research simulated scenarios where an autonomous vehicle approached merge points while interacting with multiple vehicles exhibiting varying levels of cooperation.

Despite the advancements in single-agent reinforcement learning, these approaches have a fundamental limitation when applied to on-ramp merging scenarios. By controlling only one agent and treating other vehicles as part of the static environment, they fail to capture the inherent need for coordination between multiple vehicles, particularly between those on the main highway and those merging from the ramp. To address these shortcomings and fully leverage the potential of autonomous vehicles in complex traffic scenarios, researchers have turned to a more comprehensive paradigm: MARL.

1.2.2.2 Cooperative Multi-Agent Reinforcement Learning (MARL)

As research in on-ramp merging has progressed, the limitations of single-agent approaches have become apparent, leading to the adoption of MARL. MARL offers a more comprehensive framework for addressing the inherently collaborative nature of on-ramp merging scenarios. The operational logic of MARL, as depicted in Figure 1.2, involves multiple agents (in this case, CAVs) interacting with a shared environment. Each agent performs actions based on its observed states, and the combined actions of all agents influence the environment's evolution. This cycle of observation, action, and environmental response continues until a terminal state is reached, allowing the system to learn optimal strategies for complex, multi-vehicle interactions. Normally, the CAVs on-ramp merging problem always modeled as cooperative MARL problem (Sumanth Nakka, Chalaki, and Malikopoulos 2022), (D. Chen et al. 2023), (Y. Hu et al. 2019), where all agents share the unified reward $R_1 = R_2 = \cdots = R_N = R$.

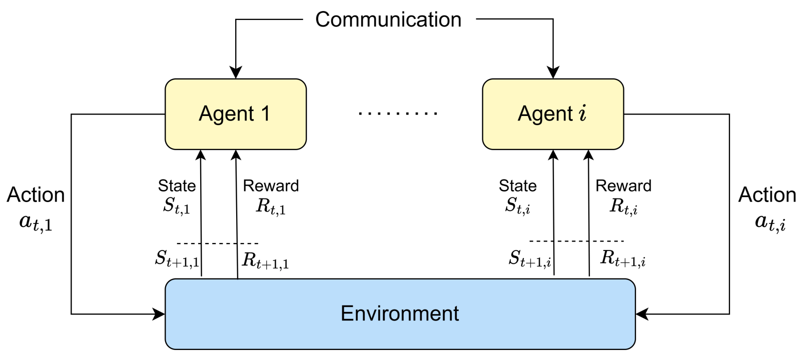
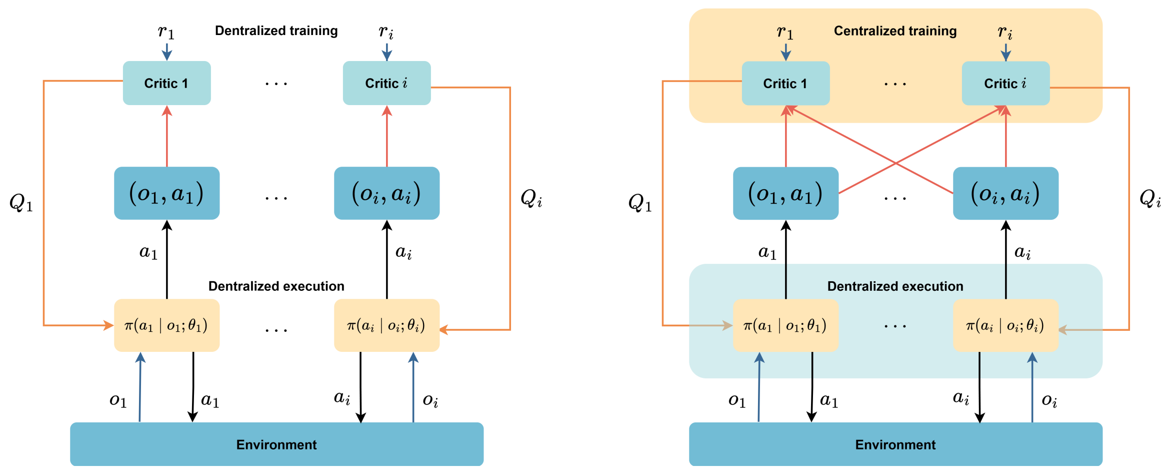


Figure 1.2: Illustration of MARL

However, in the cooperative MARL, credit assignment is crucial as algorithm updates rely on a centralized critic and consider unified rewards. When multiple agents collaborate to optimize a shared reward, accurately determining each agent's individual contribution becomes problematic. Sometimes, bad credit assignment would cause the lazy agents problem which would deteriorate the convergence performance (Du et al. 2023).

To address the challenge of credit assignment in MARL, existing methods in the realm of CAVs on-ramp merging can be broadly classified into two principal approaches: Decentralized Training with Decentralized Execution (DTDE) and Centralized Training with Decentralized Execution (CTDE). Table 1.1 summarizes key papers on Cooperative MARL for CAVs in on-ramp merging, detailing algorithms,



(a) Decentralized training with decentralized execution

(b) Centralized training with decentralized execution

Figure 1.3: Comparison of DTDE and CTDE

credit assignment approaches, action and observation spaces, and reward structures. The DTDE approach, shown in Figure 1.3a, aims to accurately capture the individual contributions of different agents by either utilizing local rewards directly or decomposing global rewards into local components, rather than relying on a unified reward system. (D. Chen et al. 2023) developed an decentralized MAA2C framework that features a priority-based safety supervisor. Their approach integrates action masking and parameter sharing to foster inter-agent cooperation. In addition, they used the local reward to capture the credit assignment. (Toghi et al. 2021) focused on developing cooperative and altruistic behaviors in CAVs using decentralized MAA2C. Their research simulates CAVs learning to travel cooperatively with each other and learn to yield to merging HDVs, with a particular emphasis on fostering altruistic behaviors in mixed traffic environments. They proposed a novel reward structure that encourages CAVs to consider the interests of other vehicles, promoting a more harmonious traffic flow. Their papers use decentralized reward structures where each agent optimizes its own reward function which is related to credit assignment. (Valiente et al. 2022) proposes a decentralized MARL framework for training cooperative CAVs in mixed-autonomy traffic. The approach uses a 3D Convolutional Neural Network with a safety prioritizer and a novel decentralized reward function that accounts for social utility. The framework addresses credit assignment through mechanisms like decentralized rewards, Social Value Orientation, semi-sequential training, and experience replay. Although the DTDE utilizes decentralized reward or local reward capturing the credit assignment, to mitigate the issue of non-stationarity (Hernandez-Leal, Kartal, and Taylor 2019), DTDE methods typically require training different agents either in isolation or in predetermined sequences (Irshayyid, J. Chen, and Xiong 2024) leading to inefficient training process.

In contrast to DTDE, the CTDE paradigm has emerged as a promising approach to address the non-stationarity challenge inherent in multi-agent systems. This framework, shown in Figure 1.3b, aims to leverage the benefits of centralized information during the training phase while maintaining the decentralized nature of execution. In the field of on-ramp merging for CAVs, the CTDE framework has gained significant traction. (Sumanth Nakka, Chalaki, and Malikopoulos 2022) introduced a comprehensive framework using MADDPG approach that employs multi-objective optimization, simultaneously considering energy efficiency and travel time for which there is not a mechanism geared for credit assignment. (Y. Hu et al. 2019) introduced the IDAS model, which employs a modified COMA method combined with curriculum learning. They used a counterfactual baseline in their centralized critic to address the credit assignment issue. To determine an agent's impact on team performance, the system evaluates the disparity between two Q-values: one reflecting the full team's actions including the agent, and another excluding that agent's specific contribution.

Although various methods have been implemented, this review still reveals a significant gap: the absence of value decomposition methods in cooperative on-ramp merging research. Value function decomposition method is specifically designed to tackle the problem of credit assignment in cooperative

Table 1.1: Cooperative MARL CAVs On-ramp Merging

Reference	Algorithm	Paradigm	Credit Assignment	Action	Observation	Reward
(D. Chen et al. 2023)	Decentralized MAA2C	DTDE	local reward	(decelerate, keep lane, accelerate, turn left, turn right)	(speed, position)	collision & speed & headway & merging waiting
(Toghi et al. 2021)	Decentralized MAA2C	DTDE	decentralized reward	(decelerate, keep lane, accelerate, turn left, turn right)	multi-channel speed position information	speed, lane change, altruistic
(Valiente et al. 2022)	DDQN	DTDE	decentralized reward	(decelerate, keep lane, accelerate, turn left, turn right)	multi-channel dynamics information	egoistic and altruistic reward
(Sumanth Nakka, Chalaki, and Malikopoulos 2022)	MADDPG	CTDE	-	acceleration	(position, speed)	rear-end safety & lateral safety & vehicle energy consumption
(Y. Hu et al. 2019)	COMA	CTDE	counterfactual baseline	(high decelerate, decelerate, maintain-speed, accelerate, high accelerate)	(road priority, driver type, current lane occupancy, other lane occupancy)	finish & collide & flow& impede

MARL. In value function decomposition method, to ensure effective coordination, a centralized action-value function, Q_{tot}, is developed, encapsulating the collective benefits achievable by the entire system. However, to direct individual agents with a decentralized policy, it is crucial to accurately attribute portions of Q_{tot} to each agent's contribution. The core principle of value function decomposition lies in breaking down a joint value function into separate value functions for each agent, thereby isolating and recognizing the unique contributions of each agent towards cooperative objectives. The pioneer work is VDN which employs a linear combination of individual Q value functions, which is suitable for simpler scenarios (Sunehag et al. 2018). To address more complex problems, QMIX introduces a monotonicity constraint that enhances the representational capacity of the model (Rashid, Samvelyan, C. S. D. Witt, et al. 2020). Based on QMIX, different kinds of constraints are investigated (Son et al. 2019), (Rashid, Farquhar, et al. 2020). Notably, the study by (J. Hu et al. 2023) demonstrates that through specific code-level fine-tuning techniques, QMIX can be enhanced to achieve state-of-the-art performance in the StarCraft Multi-agent Challenge (SMAC) task (Samvelyan et al. 2019), which shows the advantage over other kinds of constraints in the mixing network combination.

1.2.2.3 Safety-related Mechanisms

In mixed-traffic on-ramp merging scenarios, ensuring collision-free operations is paramount. Researchers have developed various safety-enhancing approaches within MARL frameworks to address this critical requirement.

(D. Chen et al. 2023) introduced a rule-based safety supervisor that projects vehicle trajectories for future time steps based on the current policy, ensuring that only safe actions are executed. To improve learning efficiency and safety simultaneously, (Y. Hu et al. 2019) implemented an action masking mechanism. This technique prevents autonomous vehicles from exploring implausible or unsafe states, effectively reducing the action search space while enhancing overall safety. By limiting exploration to sensible and safe actions, this method accelerates the learning process and improves the robustness of the learned policies.

1.3 Research Questions

This study aims to address the identified gap in applying value function decomposition methods to cooperative on-ramp merging. The main research question and its sub-questions are as follows:

- *How can value function decomposition, specifically QMIX, be effectively implemented in cooperative on-ramp merging to enhance traffic efficiency and safety?*

– How should the observation space, action space, and reward function be designed to balance safety and efficiency in on-ramp merging scenarios?

– What impact do improvement techniques, such as Q(λ) return, have on QMIX's performance (efficiency and safety) in on-ramp merging tasks?

– How can an effective action masking mechanism be designed to improve safety and learning efficiency in QMIX for on-ramp merging?

– How does the proposed algorithm compare to other state-of-the-art MARL methods in terms of credit assignment in training and overall performance (efficiency and safety) in on-ramp merging scenarios?

1.4 Structure of the Report

The remainder of this thesis is structured as follows: chapter 2 presents the necessary preliminaries, including basic concepts of reinforcement learning, Q-learning, deep Q-learning, and the QMIX algorithm. chapter 3 details our methodology, formulating the problem and introducing the proposed QMIX-QLambdaM algorithm. chapter 4 describes the experimental setup, presents results, and provides a comprehensive analysis of both learning and testing phases. chapter 5 discusses the findings, acknowledges limitations suggests directions for future research, addresses the research questions, and concludes the thesis. chapter 6 offers recommendations for road authorities and vehicle manufacturers based on our research findings.

2 Preliminaries

2.1 Basic Concepts of Reinforcement Learning

- ***Agent:*** The agent functions as the decision-maker in reinforcement learning (RL). It perceives the environment's state s_t and selects actions a_t according to a policy π. The primary objective is to maximize the aggregate reward over the entire duration.
- ***Environment:*** The environment defines the context in which the agent operates. It reacts to the actions of the agent a_t by updating the states to s_{t+1} and provides rewards r_t, which are crucial for steering the learning algorithm.
- ***State and Action:*** The state represents the current condition of the agent within the environment. Formally, the state at any given time t is expressed as $s_t \in S$, where S denotes the *state space*. From any given state, the agent may take an action $a_t \in A$, where A is the collection of possible actions from state s_t, known as the *action space*. The action space can vary across different states.
- ***State Transition:*** Actions by the agent trigger transitions from one state to another, a process referred to as state transition. For instance, if the agent is in state s_1 and opts for action a_2, it will transition to state s_2. This transition can be depicted mathematically as:

  $$s_1 \xrightarrow{a_2} s_2 \tag{2.1}$$

  Additionally, the process of state transition can be modeled using conditional probabilities, as shown in the following example:

  $$p(s_2 \mid s_1, a_2) = 1 \tag{2.2}$$

  This indicates that selecting action a_2 while in state s_1 guarantees a transition to state s_2.
- ***Policy:*** The policy, symbolized by π, dictates the agent's choice of action based on the current state. It assigns actions to states, possibly in a stochastic manner, as denoted by $\pi(a \mid s)$ which represents the likelihood of choosing action a in state s. An example of such a policy for state s_1 is:

  $$\begin{aligned} \pi(a_1 \mid s_1) &= 0 \\ \pi(a_2 \mid s_1) &= 1 \\ \pi(a_3 \mid s_1) &= 0 \\ \pi(a_4 \mid s_1) &= 0 \\ \pi(a_5 \mid s_1) &= 0 \end{aligned} \tag{2.3}$$

  This implies that the probability of executing action a_2 at state s_1 is certain, while other actions have zero probability.

- ***Reward:*** A reward r serves as feedback from the environment assessing the efficacy of an action a executed from a specific state s. This function, denoted as $r(s, a)$, can yield positive, negative, or zero values. Varying rewards influence the policy that the agent ultimately learns, encouraging actions associated with positive rewards and discouraging those linked to negative outcomes.

- ***Trajectories and Episodes:*** A *trajectory* consists of a sequence of state-action-reward events. For instance, a trajectory from s_1 to s_9 might appear as:

$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9. \tag{2.4}$$

  If a trajectory reaches a *terminal state*, it is termed an *episode*. Thus, the trajectory Equation 2.4 qualifies as an episode.

- ***Returns:*** The *return* of a trajectory, such as Equation 2.4, is calculated as the cumulative sum of rewards obtained along that path.

$$return = 0 + 0 + 0 + 1 = 1 \tag{2.5}$$

  Returns, also known as *total rewards* or *cumulative rewards*, are used to assess the effectiveness of the corresponding policy. However, in practical applications, the *discounted return* is more commonly used to address the potential infinity of episode lengths, introducing a discount rate $\gamma \in (0, 1)$. This is particularly relevant in real-world problems where episodes can be exceedingly long or effectively infinite. The discounted return is computed as the sum of discounted rewards:

$$discounted\ return = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 \cdots \tag{2.6}$$

  Additionally, the value of γ plays a crucial role in determining the policy's focus. A γ close to 1 encourages the agent to consider long-term rewards more heavily, making the policy more far-sighted. Conversely, a γ closer to 0 makes the agent prioritize immediate rewards, rendering the policy more short-sighted.

- ***Interaction Process:*** The dynamic between the agent and the environment in RL is typically framed as a Markov Decision Process (MDP). Each time step t involves the agent observing the current state s_t, choosing an action a_t per its policy π, receiving a reward r_{t+1}, and transitioning to a new state s_{t+1} as determined by the environment's rules. This cycle persists, with the agent continually refining its policy based on the feedback received, aiming to maximize the overall reward.

- ***Value Function:*** The value function, denoted by $V(s)$, quantifies the expected cumulative reward from a given state s when adhering to a specific policy π. It is formally expressed as:

$$V_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

  where γ is the discount factor, constrained between 0 and 1 $(0 \leq \gamma \leq 1)$. This factor weighs the significance of future rewards against immediate gains, effectively modulating the preference for short-term versus long-term benefits. The function helps in understanding how beneficial it is to be in a particular state, considering all future possibilities under the policy π.

- ***Action Value Function:*** Commonly referred to as the Q-function, $Q(s, a)$ estimates the total expected return from selecting an action a in state s, followed by continuous adherence to policy π. Its definition is:

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right],$$

  This function extends the concept of the value function by incorporating the immediate action, offering a more granular view of the decision-making process in RL. It captures the expected effectiveness of each action within every state, thus serving as a critical component in both evaluating and optimizing policies.

2.2 Q Learning & Deep Q Learning

2.2.1 Temporal Difference (TD) Learning

The objective of TD learning is to approximate the state value function $v_\pi(s)$ for all states $s \in S$, under a given policy π. Consider a trajectory of experience samples $(s_0, r_1, s_1, ..., s_t, r_{t+1}, s_{t+1}, ...)$, generated by adhering to policy π. The TD learning algorithm estimates the state values for these samples using the following update rule:

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - (r_{t+1} + \gamma(s_{t+1}))\right], \tag{2.7}$$

$$v_{t+1}(s) = v_t(s), \quad \text{for all } s \neq s_t \tag{2.8}$$

In these equations, $t = 0, 1, 2, ..., t$ represents the time steps. The term $v_t(s_t)$ denotes the current estimate of $v_\pi(s_t)$ at time t. The learning rate at time t for state s_t is $\alpha_t(s_t)$. During each step t, only the value of the state s_t that was visited is updated, while values for all other states remain unchanged. Annotated further, the equation 2.7 can be broken down as:

$$\underbrace{v_{t+1}(s_t)}_{\text{new estimate}} = \underbrace{v_t(s_t)}_{\text{current estimate}} - \alpha_t(s_t)\left[\overbrace{v_t(s_t) - (\underbrace{r_{t+1} + \gamma v_t(s_{t+1})}_{\text{TD target } \bar{v}_t})}^{\text{TD error } \delta_t}\right], \tag{2.9}$$

where $r_{t+1} + \gamma v_t(s_{t+1})$ forms the TD target $\bar{v}_t$, which is the objective that v_t attempts to achieve. The term δ_t is referred to as the TD error, representing the difference the algorithm aims to minimize. The new estimate $v_{t+1}(s_t)$ is an adjustment of the current estimate $v_t(s_t)$, influenced by the magnitude of the TD error δ_t.

In TD learning algorithms, the tabular method is commonly used to track and systematically update estimated values for each state or state-action pair. We can represent this tabular method for TD learning as shown in Table 2.1, where each state has an associated estimated value updated upon visitation.

Table 2.1: Estimated values for states

State	s_1	s_2	$\cdots$	s_n
Estimated value	$\hat{v}(s_1)$	$\hat{v}(s_2)$	$\cdots$	$\hat{v}(s_n)$

2.2.2 TD Learning of Action Values: SARSA

SARSA, similar to Temporal Difference (TD) learning, is tailored to estimate action values directly, whereas TD learning typically focuses on state values. Under a defined policy π, SARSA aims to calculate the action value $Q_\pi(s, a)$ for each state-action pair. Consider a series of experience samples generated following policy π: $(s_0, a_0, r_1, s_1, a_1, ..., s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, ...)$. The following equations articulate the SARSA algorithm's method for updating action values:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[Q_t(s_t, a_t) - (r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}))\right], \tag{2.10}$$

$$Q_{t+1}(s, a) = Q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t) \tag{2.11}$$

In these updates, $t = 0, 1, 2, ...$ indexes the time steps, and $\alpha_t(s_t, a_t)$ denotes the learning rate for the state-action pair (s_t, a_t) at time t. The estimate $Q_t(s_t, a_t)$ approximates the action value under policy π at that particular time. Only the value of the state-action pair visited at time t, (s_t, a_t), is updated, while all other action values remain unchanged.

The nomenclature ”SARSA” reflects the algorithm's dependency on the sequence of state-action-reward-state-action elements, hence the acronym SARSA. This method diverges from TD learning by focusing on action value estimations rather than state values, making it particularly suitable for policies where actions are explicitly evaluated.

2.2.3 TD Learning of Optimal Action Values: Q-learning

Unlike Sarsa, which adheres to the current policy to determine its learning path, Q-learning updates its action-value estimates based on the maximum potential future reward, irrespective of the action taken. This makes Q-learning an off-policy learner, which essentially learns the optimal policy even as it behaves differently. The Q-learning update rule is formulated as follows:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[Q_t(s_t, a_t) - (r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a))\right] \tag{2.12}$$

$$Q_{t+1}(s, a) = Q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t) \tag{2.13}$$

In this framework, $t = 0, 1, 2, ...$ denotes the time steps, $Q_t(s_t, a_t)$ represents the estimated action value at time t, and $\alpha_t(s_t, a_t)$ is the learning rate for the state-action pair (s_t, a_t). Each update modifies only the action value for the state-action pair encountered, with all other values remaining constant.

The critical distinction between Q-learning and Sarsa lies in their respective TD targets: Q-learning's TD target is defined by the expression $r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a)$, indicating that it seeks the highest possible reward from the next state, maximizing the expected utility without considering the specific next action. This contrasts with Sarsa, which uses the actual next action's value $r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1})$ for updates, closely tying the learning to the chosen policy path. Furthermore, in terms of data requirements per iteration, Q-learning needs less information: while SARSA requires the next state-action pair (s_{t+1}, a_{t+1}) along with the reward, Q-learning requires only the next state s_{t+1} and the reward r_{t+1}.

Similar to TD learning, Q-learning can be implemented using a tabular approach. The key distinction lies in that Q-learning estimates the value of state-action pairs, rather than states alone.

Table 2.2: Estimated values for state-action

Estimated Q-value	a_1	a_2	$\cdots$	a_n
s_1	$\hat{Q}(s_1, a_1)$	$\hat{Q}(s_1, a_2)$	$\cdots$	$\hat{Q}(s_1, a_t)$
s_2	$\hat{Q}(s_2, a_1)$	$\hat{Q}(s_2, a_2)$	$\cdots$	$\hat{Q}(s_2, a_t)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
s_3	$\hat{Q}(s_3, a_1)$	$\hat{Q}(s_3, a_2)$	$\cdots$	$\hat{Q}(s_3, a_3)$

2.2.4 Deep Q-learning

However, when dealing with large or infinite state and action spaces, the tabular method faces challenges in efficiently retrieving and storing data. An alternative approach is to use function approximation, where the estimated values in the table are treated as points on a curve, such as a linear function expressed as:

$$\hat{v}(s, w) = as + b = \underbrace{[s, 1]}_{\phi^T(s)} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{w} = \phi^T(s)w.$$

In this formula, $\hat{v}(s, w)$ approximates $v_\pi(s)$ using the feature vector $\phi(s) \in \mathbb{R}^2$, thereby enhancing storage efficiency and enabling generalization. Unlike the tabular method which updates values only during visits to corresponding states, function approximation updates the parameters influencing all data samples, thereby updating even unvisited states.

Integrating deep neural networks into Q-learning leads to the development of deep Q-learning, as outlined by (Mnih et al. 2015). The primary goal of deep Q-learning is to minimize the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t, a_t, r_{t+1}, s_{t+1}) \sim D}\left[\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \tag{2.14}$$

where D is the experience replay buffer, and θ^- represents the periodically updated weights of the target network, distinct from the current weights θ of the value network.

- ***Experience Replay:*** The idea of experience replay is to ensure that after collecting some experience samples $(s_t, a_t, r_{t+1}, s_{t+1})$, we do not use them in the order they were collected. Instead, we store them in the $replay\ buffer : D$. Every time we update the value network, we can draw a mini-batch of experience samples from the replay buffer $uniformly$ to break the correlation between the samples in the sequence to satisfy the uniform distribution.
- ***Target Network:*** The idea for this technique is that when we use gradient descent to update the parameter θ of value network, we can keep the parameter of the target network θ^- as the same (for a short period), which is helpful for stabilizing the training.

2.3 QMIX

The decomposition method of the value function is designed to address the challenge of credit assignment by factorizing the joint Q-value function into a combination of individual Q-value functions. This factorization distinguishes the unique contributions of each agent, which is realized by Individual-Global-Max (IGM) principle (Son et al. 2019). This principle ensures that the optimal joint action, obtained through the global maximization of the team's Q function value, aligns with the aggregation of Q function values that have been maximized individually, which is mathematically expressed in 2.15.

$$\arg\max_{a} Q_{tot}(\tau, a) = \begin{pmatrix} \arg\max_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \arg\max_{a_N} Q_N(\tau_N, a_N) \end{pmatrix} \tag{2.15}$$

In QMIX, to ensure the condition of Equation 2.15 hold, additionally, a constraint Equation 2.16 is imposed on both the collective and individual value functions to guarantee their monotonicity, ensuring that the integrated value function consistently reflects the aggregated impact of each agent's actions.

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A \tag{2.16}$$

The architecture of the QMIX algorithm is shown in Figure 3.3, which features two critical components: the agent network and the mixing network. The agent network receives as input the individual observations and previous actions of each agent. Its configuration employs recurrent neural networks (RNNs) to enable agents to leverage their complete history of actions and observations. Essentially, the agent network generates an estimated action value Q_i for agent i, which is subsequently relayed to the mixing network. The mixing network incorporates the overall state of the environment along with the estimated action values as input. The parameters of the mixing network are generated through a set of dedicated hypernetworks. Each hypernetwork receives the state s as input and produces weights for one layer of the mixing network. These hypernetworks have a relatively simple structure, consisting of a single linear layer followed by an absolute value activation function. The purpose of using an absolute value activation is to ensure that the generated mixing network weights are all non-negative. The output of each hypernetwork is a vector, which is subsequently reshaped into a matrix of appropriate dimensions. The process for generating bias terms is similar to that of weights, but without the non-negativity constraint. Notably, the final bias term is produced by a slightly more complex hypernetwork, which incorporates two layers with a ReLU non-linearity between them. The primary goal is to optimize the network by minimizing the specified loss function.

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(r + \gamma \max_{u'} Q_{tot}(\tau', u', s'; \theta') - Q_{tot}(\tau, u, s; \theta) \right)^2 \right] \tag{2.17}$$

The loss function, as referenced in Equation 2.17, aligns with the traditional DQN algorithm's framework, with the distinction that the Q value in this context is a cumulative Q_{tot} value. Throughout the training phase, every agent implements an ϵ-greedy strategy, where ϵ regulates the equilibrium between exploration and exploitation for each agent, gradually decreasing as training progresses.

3 Methodology

3.1 Methodological Challenges

3.1.1 Dec-POMDP

In addressing the challenge of multi-vehicle driving, we could recognize that each agent operates with limited perception and uncertain action outcomes. To model the *cooperative multi-agent task*, the problem is specifically modeled as a decentralized partially observable Markov decision process (Dec-POMDP) model. This model is succinctly represented by the tuple:

$$G = \langle S, A, P, r, Z, O, n, \gamma \rangle$$

Here, $s \in S$ denotes the actual state of the environment, with the Dec-POMDP framework accounting for situations where each agent receives its own incomplete observation $z \in \mathcal{Z}$, derived from the observation function $O(s, a) : S \times A \rightarrow [0, 1]$. Within each time step, an agent i from the set $N \equiv 1, \ldots, n$ selects an action $a_i \in A_i$ guided by its policy $\pi_i (a_i \mid \tau_i) : \mathcal{T} \times \mathcal{A} \rightarrow [0, 1]$, where $\tau_i \in \mathcal{T} := (\mathcal{Z} \times \mathcal{A})^*$ represents the agent's history of actions and observations. These individual actions combine into a collective action $a \in \mathcal{A} \equiv A'$, leading to the next state according to the state transition function $P(s' \mid s, a) : S \times \mathcal{A} \times S \rightarrow [0, 1]$. A unified reward function $r(s, a) : S \times \mathcal{A} \rightarrow \mathbb{R}$ benefits all agents, with the action-value function defined as $Q^{\pi} (s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}} [R_t \mid s_t, a_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ represents the reward accumulated over time, discounted at rate γ. While the training process is conducted in a centralized manner, allowing the learning algorithm to utilize both the collective action-observation histories τ and the overall state s, the execution phase is decentralized.

3.1.2 Defining State, Action, and Reward Structures

- **State and Observation Space:** The state space in reinforcement learning for autonomous driving should balance essential information with computational efficiency. It typically includes the ego vehicle's kinematics and data on surrounding traffic. Our approach focuses on immediately adjacent vehicles to maintain training efficiency while capturing crucial environmental dynamics (Irshayyid, J. Chen, and Xiong 2024). Each agent i receives observations as a matrix o_i with dimensions $N_i \times W$. N_i represents observable vehicles within 150 meters (Yu et al. 2020), usually up to six: front, rear, left front, left rear, right front, and right rear. W encompasses various vehicle attributes:
  - $ispresent$: This binary variable denotes whether a vehicle occupies this feature position within the observable surroundings.
  - x : Represents the relative horizontal displacement of surrounding vehicles with respect to the ego vehicle. **For the ego vehicle itself, this value is set to zero to optimize training efficiency. While this information is primarily relevant for evaluating lane change rewards (discussed later), the pertinent data is captured in the last action feature $\bar{u}$.**

  - y : Indicates the relative vertical displacement of surrounding vehicles in relation to the ego vehicle. For the ego vehicle, this is set to 0, similar to x.
  - vx : Represents the relative horizontal speed of surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the absolute horizontal velocity.
  - vy : Denotes the relative vertical speed of surrounding vehicles in relation to the ego vehicle. For the ego vehicle, this is the absolute vertical velocity.
  - cos_h : Represents the cosine of the relative heading angle for surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the cosine of its absolute heading.
  - sin_h : Indicates the sine of the relative heading angle for surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the sine of its absolute heading.
  - $\bar{u}$: Denotes the action executed by the agent in the previous time step.

  The comprehensive state space is formed by amalgamating the observations of all individual agents, denoted as $s = [o_i]_{i=1}^{N}$, where N represents the total number of controlled CAVs.

- **Action Space:** The range of actions available to each agent comprises several high-level strategic maneuvers such as $lane\ change\ left$, $lane\ change\ right$, $maintain\ lane$, $accelerate$, and $decelerate$ for training efficiency without sacrificing the essence of driving behavior. These strategies are implemented through a lower-level control system, which adjusts the steering and throttle based on the selected action. The collective action space for all the CAVs in the system is a composite of the individual actions, represented as $A = A_1 \times A_2 \times \cdots \times A_N$.

- **Reward Function:** The reward function R_i is crucial in guiding agent behavior towards desired outcomes. Our design prioritizes two primary objectives:

  1. Safety: Encouraging collision avoidance
  2. Efficiency: Promoting higher speeds for optimal travel

  These main indicators form the core of our reward structure. Additionally, the function incorporates secondary factors that support these primary goals, including:

  - Maintaining safe distances from surrounding vehicles
  - Minimizing unnecessary lane changes
  - Discouraging prolonged waiting in the merging lane

  While these secondary aspects contribute to overall performance, the emphasis remains on collision avoidance and speed optimization. The reward r_i for each agent i is formulated to reflect this prioritization:

$$r_i = w_c r_c + w_s r_s + w_{ttc} r_{ttc} + w_l r_l + w_m r_m \tag{3.1}$$

  where w_c, w_s, w_{ttc}, w_l, and w_m are positive weights for the penalty of collision, speed reward, time-to-collision (TTC) evaluation, penalty for change of lanes, and the driving task reward. The five performance indicators are defined as follows in more detail:

  - the r_c of the agent i is set to be -1 if the vehicle has a collision, otherwise it is 0.
  - the r_s is defined as

$$r_s = \min\left\{\frac{v_t - v_{\min}}{v_{\max} - v_{\min}}, 1\right\} \tag{3.2}$$

    The settings for $v_{\min} = 10m/s$ and $v_{\max} = 30m/s$ are based on the configurations described in (D. Chen et al. 2023). These parameters were chosen by considering two key sources: recommendations from the (US Department of Transportation 2018) and empirical data from the Next Generation Simulation (NGSIM) dataset (Thiemann, Treiber, and Kesting 2008). The upper limit aligns with the transportation authority's suggested speed range $(20 - 30m/s)$. At the same time, the lower bound takes into account the minimum speeds observed in real-world traffic scenarios, as captured in the NGSIM data (minimum speed at $6 - 8m/s$).

- As the vehicle can observe up to 6 surrounding vehicles, it is crucial to determine the appropriate number of vehicles to consider in the TTC evaluation. The reward associated with TTC, r_{ttc}, is defined in Equation 3.3. Furthermore, a variable N represents the set of evaluation vehicles, ordered from the smallest to the largest TTC. **The possible values for N range from 1 to 6, with the optimal value to be determined experimentally. The results of this determination are detailed in chapter 4.** The calculation of the TTC value follows the methodology described in (Jiao 2023), which accounts for both longitudinal and lateral TTCs. The evaluation function for the TTC value is structured as follows: the safe TTC threshold, TTC_{safe}, is set at $2s$ (Kruber et al. 2019). Since TTC values of ∞ and -1 are not applicable within the function, these are substituted with $12s$ and $1e-9$, respectively (Kruber et al. 2019). The corresponding trend for this function is depicted in Figure 3.1, demonstrating that when the TTC is below the safe threshold, the reward decreases significantly.

$$r_{ttc} = \frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{TTC_n}{TTC_{safe}}\right), \quad TTC_n \in [1e-9, 12], \quad n \in N \tag{3.3}$$

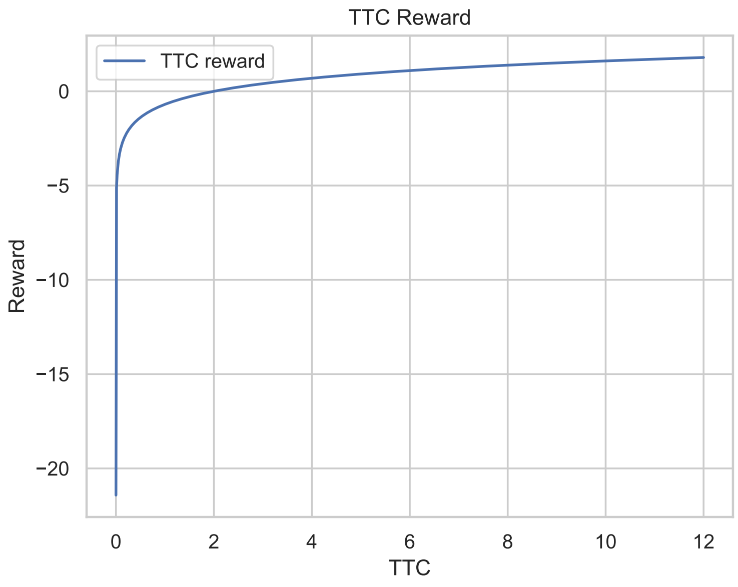


Figure 3.1: TTC Reward

- For agent i, the lane change reward, r_l, is set to -1 if the vehicle is not merging vehicle, otherwise, the reward is zero.
- The merging task reward for agent i is formulated as a function of the distance traveled on the ramp, denoted as x. This cost is designed to encourage timely merging and is defined as:

$$r_m = \frac{1}{40000}x^2 - \frac{1}{100}x, \quad x \in [0, l_{ramp}] \tag{3.4}$$

  where l_{ramp} represents the total length of the merging ramp. Figure 3.2 illustrates the relationship between the merging cost and the distance traveled. This quadratic function ensures that the cost increases more rapidly as the vehicle approaches the end of the merging lane, thereby discouraging prolonged waiting in the merging area.

3.2 QMIX-QLambdaM: Improved QMIX

In this section, we present QMIX-QLambdaM, our proposed enhancement to the QMIX algorithm for cooperative on-ramp merging scenarios. QMIX-QLambdaM incorporates two key improvements: the

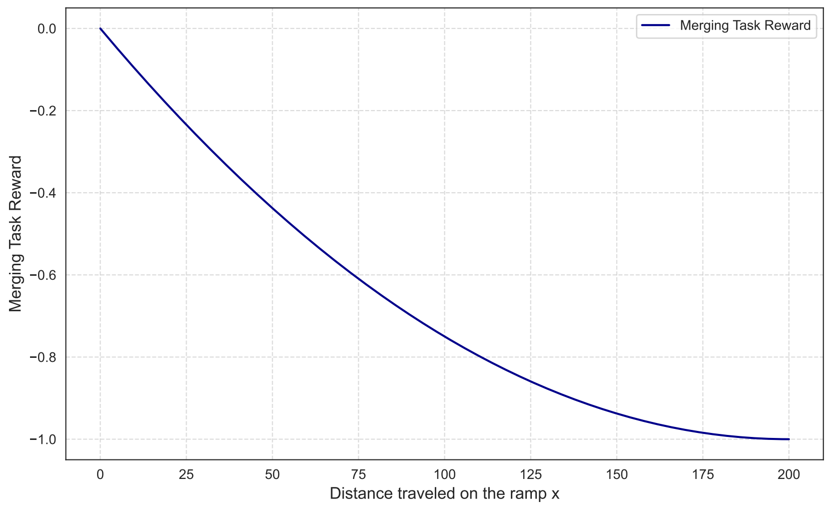


Figure 3.2: Merging task reward

$Q(\lambda)$ return for more efficient value estimation, and an action masking mechanism for safer action selection which is detailed in Figure 3.3.

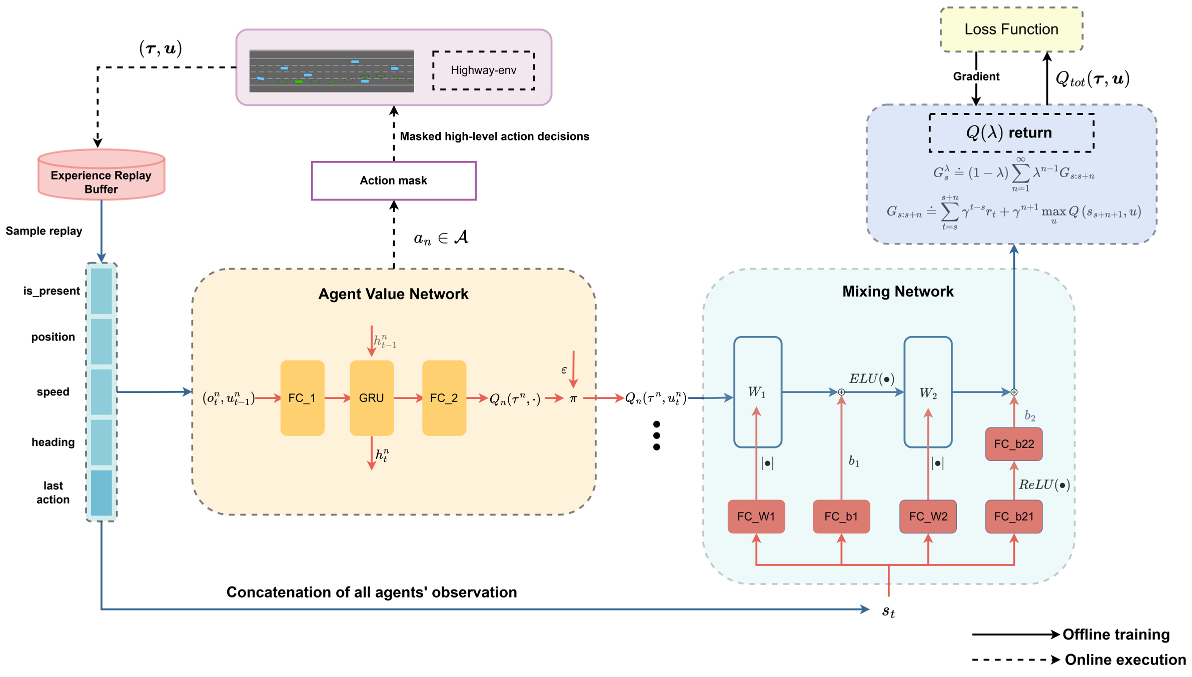


Figure 3.3: The structure of QMIX-QLambdaM

3.2.1 Q(λ) Return

In the section 2.2, it is noted that TD learning is employed to compute the state value $V(s)$ and the action value $Q(s, a)$. TD learning allows for the immediate update of state/action values upon receiving an experience sample at each time step, making it particularly suitable for continuous tasks. However, TD learning introduces a higher degree of bias compared to the actual values since each update relies on the previous estimate, necessitating an initial guess of these values.

Conversely, the MC method computes the state/action value by leveraging the total discounted return from each complete episode, making it well-suited for episodic tasks. As a result, MC must wait until the conclusion of an episode to update values, leading to less bias as it directly utilizes the actual returns without requiring an initial guess. However, compared to TD learning, MC experiences higher variance issues. For instance, to estimate the action value $q_\pi(s_t, a_t)$ using MC, samples from the sequence $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$ are needed, whereas in Sarsa, only three variables are required: $R_{t+1}, S_{t+1}, A_{t+1}$.

Given the advantages and drawbacks of both the MC and TD methods, a family of approaches called eligibility traces has been developed to balance the MC returns with the one-step TD guesses, thereby accelerating the learning process. This includes methods such as TD(λ) and Peng's Q(λ), as referenced in (J. Hu et al. 2023). These techniques, applicable to value-based and episodic models like QMIX, introduce a hybrid form of return calculation, the function of TD(λ) is as shown:

$$\begin{aligned} G_s^\lambda &\doteq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{s:s+n}, \\ G_{s:s+n} &\doteq \sum_{t=s}^{s+n} \gamma^{t-s} r_t + \gamma^{n+1} V\left(s_{s+n+1}, u\right). \end{aligned} \tag{3.5}$$

The mechanism of TD(λ) is detailed in Figure 3.4. The first column aligns with the classic TD learning. The last column aligns with the MC return. In between there are different trajectory with different steps of return. For each column, the equations below indicate the corresponding weights which in total are 1.

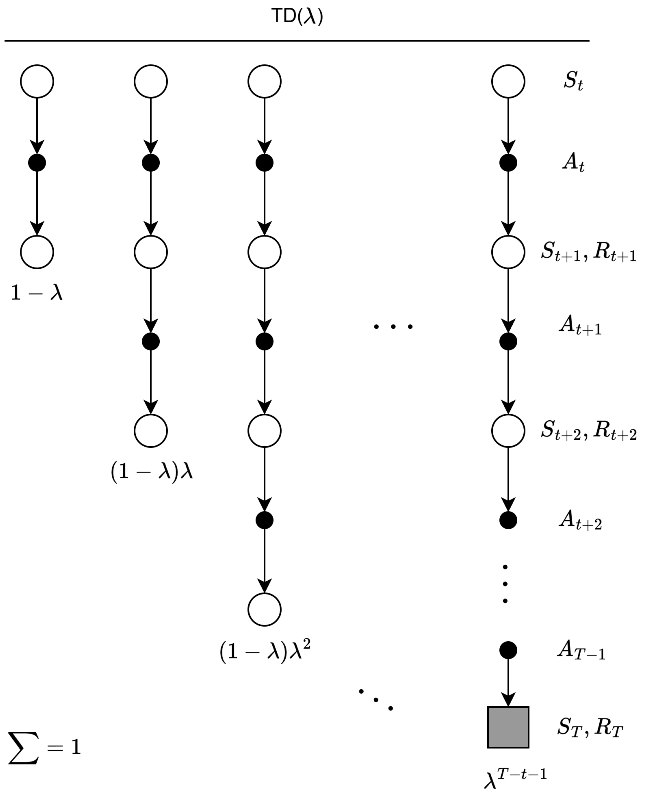


Figure 3.4: TD(λ)

Peng's Q(λ) modifies this by substituting the V value of the next state with the maximum Q value, further detailed in Equation 3.6:

$$G_{s:s+n} \doteq \sum_{t=s}^{s+n} \gamma^{t-s} r_t + \gamma^{n+1} \max_u Q\left(s_{s+n+1}, u\right), \tag{3.6}$$

where $G_{s:s+n}$ represents the discounted return from state s to state n. To illustrate, consider an episode length of 4 with $s = 0$, the returns are calculated as follows:

$$\begin{aligned}
G_{s:s} &= r_0 + \gamma \max_a Q(s_1, a), \quad n = 0, \\
G_{s:s+1} &= r_0 + \gamma r_1 + \gamma^2 \max_a Q(s_2, a), \quad n = 1, \\
G_{s:s+2} &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 \max_a Q(s_3, a), \quad n = 2, \\
G_{s:s+3} &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^4 \max_a Q(s_4, a), \quad n = 3, \\
G_{s:s+4} &= r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^4 r_4, \quad n = 4.
\end{aligned} \tag{3.7}$$

When $n = 4$, which aligns with the total episode length, the actual reward is used to compute the return. Furthermore, based on the update rule of Q-learning discussed in section 2.2, G_s^λ can be computed recursively, enhancing efficiency:

$$G_s^\lambda = r_t + \gamma[(1-\lambda) \max_a Q(s_{s+1}, a) + \lambda G_{s+1}^\lambda]. \tag{3.8}$$

Here, λ serves as the discount factor for the traces; setting λ to 0 corresponds to classic Q-learning, while setting it to 1 aligns with MC returns. **The optimal value of λ for this project will be experimentally determined, as described in chapter 4**.

3.2.2 Action Masking Mechanism

To enhance safety and accelerate training, QMIX-QLambdaM employs an action masking mechanism that constrains the action selection process to a subset of reasonable and safe actions. This approach not only prevents actions that could lead to accidents or traffic rule violations but also expedites the learning process by effectively reducing the action space. The mechanism, integral to QMIX-QLambdaM's design, is implemented during the action selection phase.

The action masking mechanism in QMIX-QLambdaM comprises two key filtering algorithms: `LongitudinalMask` and `LateralMask`, detailed in Algorithms 1 and 2, respectively. These algorithms ensure the safety of both speed change and lane change commands. It's worth noting that actions violating speed limits or road topology constraints are already masked by built-in functions in the simulator. In the pseudocode, italics represent variables, while bold text indicates algorithms and equations. The details of these algorithms and equations are provided in the Appendix B. At its core, this mechanism operates by simulating each potential action for one time step before the agent's selection, allowing for a proactive safety assessment.

In the `LongitudinalMask`, the inputs are the front vehicle and the rear vehicle. Initially, both are processed by a function called `UnsafeVehicle`, which evaluates the longitudinal distance between two vehicles to determine if further assessment is necessary. If a surrounding vehicle is deemed safe, it is set to None. Next, the activation of `AccelerateMask` and `DecelerateMask` depends on the presence of v_f and v_r, which represent the front and rear vehicles, respectively. The main function of `AccelerateMask` is to simulate, for the next time step, the ego vehicle executing $acceleration$ or $idle$, while the corresponding surrounding vehicle, depending on its type, executes its own action. If v_{surr} is a controlled vehicle, it will execute the most dangerous action relative to the ego vehicle (in this case, v_f executing $deceleration$). However, if it is a human-driven vehicle, it will follow the IDM policy. The difference between vehicle types is that for controlled vehicles, the masking mechanism will override their policy, whereas human-driven vehicles will always adhere to their policy. The simulation lasts for one step, as the action frequency is one action per time step. During the simulation, if the TTC value (note that this is one-dimensional, as the $2DTTC$ referred to in chapter 3 is computationally intensive) between the ego vehicle and the surrounding vehicle is shorter than the TTC_{safe}, the executed action for the ego vehicle will be masked out in the available action space. It's important to note that masking the $idle$ action will only occur when actions other than $acceleration$ and $idle$ are available, to prevent an empty action space. The logic is similar for the `DecelerateMask`. When both v_f and v_r are unsafe regarding the ego vehicle, their TTC values will be compared to determine which poses the greater danger to determine which mask will be activated.

In the `LateralMask`, decision-making focuses on whether to filter out actions for turning left or right. Algorithm 2 details this process. The vehicles v_{lf}, v_{lr}, v_{rf}, and v_{rr} are processed through `UnsafeVehicle` to assess their safety. Subsequently, the activation of `LeftMask` and `RightMask` depends on the presence of corresponding surrounding vehicles. This logic is more intricate compared to the `LongitudinalMask`, as identifying the most hazardous scenario for the ego vehicle depends on the relative position of the surrounding vehicles. In the `LeftMask`, for instance, one case considers a left vehicle (a CAV) not in the immediately adjacent lane, assumed to turn right. This represents the most threatening situation when the ego vehicle intends to turn left. Conversely, if the left vehicle is in the adjacent lane and is a CAV, depending on its relative longitudinal position to the ego vehicle, the most dangerous action v_{lf} and v_{lr} are $acceleration$ and $deceleration$, respectively. Unlike the longitudinal scenario, if a collision is detected during these maneuvers, the option to change lanes left is filtered out of the available actions. The logic is also similar to `RightMask`. Consequently, up to two surrounding vehicles are considered in both `LeftMask` and `RightMask`.

Algorithm 1 LongitudinalMask

```
1:  function LongitudinalMask($ego$, $v_f$, $v_r$, $available\_actions$, $t$)
2:      $v_f \leftarrow$ UnsafeVehicle($ego$, $v_f$)
3:      $v_r \leftarrow$ UnsafeVehicle($ego$, $v_r$)
4:      if $v_f \neq$ None and $v_r =$ None then
5:          $available\_actions \leftarrow$ AccelerateMask($ego$, $v_f$, $available\_actions$, $t$)
6:      end if
7:      if $v_r \neq$ None and $v_f =$ None then
8:          $available\_actions \leftarrow$ DecelerateMask($ego$, $v_r$, $available\_actions$, $t$)
9:      end if
10:     if $v_f \neq$ None and $v_r \neq$ None then
11:         $TTC_f \leftarrow$ 1DTTC($ego$, $v_f$)
12:         $TTC_r \leftarrow$ 1DTTC($ego$, $v_r$)
13:         if $TTC_f \leq TTC_r$ then
14:             $available\_actions \leftarrow$ AccelerateMask($ego$, $v_f$, $available\_actions$, $t$)
15:         else
16:             $available\_actions \leftarrow$ DecelerateMask($ego$, $v_r$, $available\_actions$, $t$)
17:         end if
18:     end if
19:     return $available\_actions$
20: end function
```

Algorithm 2 LateralMask

```
1:  function LateralMask($ego$, $v_{rf}$, $v_{rr}$, $v_{lf}$, $v_{lr}$, $available\_actions$, $t$)
2:      $v_{lf} \leftarrow$ UnsafeVehicle($ego$, $v_{lf}$)
3:      $v_{lr} \leftarrow$ UnsafeVehicle($ego$, $v_{lr}$)
4:      $v_{rf} \leftarrow$ UnsafeVehicle($ego$, $v_{rf}$)
5:      $v_{rr} \leftarrow$ UnsafeVehicle($ego$, $v_{rr}$)
6:      if $v_{lf} \neq$ None or $v_{lr} \neq$ None then
7:          $available\_actions \leftarrow$ LeftMask($ego$, $v_{lf}$, $v_{lr}$, $available\_actions$, $t$)
8:      end if
9:      if $v_{rf} \neq$ None or $v_{rr} \neq$ None then
10:         $available\_actions \leftarrow$ RightMask($ego$, $v_{rf}$, $v_{rr}$, $available\_actions$, $t$)
11:     end if
12:     return $available\_actions$
13: end function
```

4

Experiments and Results Analysis

4.1 Experiments Setup

4.1.1 Experiments Scenario and Parameter Setup

Figure 4.1 illustrates the road structure and vehicle spawn points, with the x-coordinates of these points clearly indicated. Table 4.2 provides comprehensive details about the simulation and experimental parameters. The simulation is conducted using the `highway-env` simulator (Leurent 2018). To enhance realism, position noise is added at the spawn points, introducing variability in vehicle starting positions. Vehicles on the main lane begin moving at speeds between 25-27 m/s (90-97 km/h), while those on the merging lane start between 20-22 m/s (72-79 km/h), in accordance with (Dutch Design Guidelines 2024). The simulation enforces an action frequency of one second, meaning that the agent takes one action per second of simulation time. Although vehicles may exit the depicted road structure, their dynamics continue to be computed within the simulation framework. The distribution of HDVs and CAVs is also outlined in Table 4.2. The chosen distribution of CAVs, with one CAV per lane, allows for the capture of different behaviors based on vehicle position while keeping the learning process as simple as possible to reduce computational burden. Two HDVs are placed in each main lane to enable CAVs to learn adaptive policies, as they may spawn between the two HDVs. The merging lane contains one HDV, as the number of vehicles evaluated in the TTC reward is variable and up to six, as described in chapter 3. Consequently, the merging lane contains at least two vehicles.

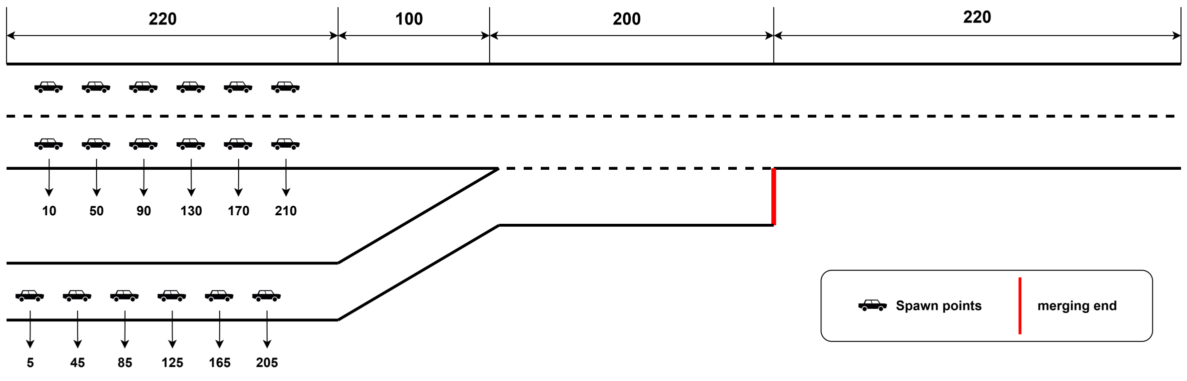


Figure 4.1: Simulation Scenario

For all training experiments mentioned in section 4.2, 15,000 episodes are executed in the training phase. The RMSprop optimizer is used consistently, with gradient clipping set to ten to prevent gradient explosion. The discount factor γ is set to 0.99 across all experiments. The target update cycle, representing the frequency of updates to the target Q network in QMIX and the target critic network in the actor-critic algorithm, is detailed in section 4.2. To ensure reproducibility, the training process starts with a random seed of 2024, incrementing by one for each subsequent episode. To monitor and record the performance of the trained policy, evaluation is conducted every 100 episodes, and the corresponding policy is saved for future reference. The evaluation phase utilizes 40 independent test seeds, completely distinct from those used in training, to prevent overfitting and ensure robust performance assessment. All training experiments are conducted on a single PC with 2.50 GHz Intel(R) Core(TM) i5-12400 CPU and NVIDIA GeForce RTX 3060 GPU.

The weights w_c, w_s, w_{ttc}, w_l, and w_d are set to 1000, 100, 100, 1, and 10, respectively. This ratio is chosen to maintain the dominance of safety-related rewards. The relative magnitudes of them are presented in Table 4.1. As shown, the collision reward and TTC reward have the highest values, followed by the speed reward. The lane change reward and merging task reward have the smallest values.

Table 4.1: Magnitude Comparison of Different Kinds of Reward

Weight	**Magnitude**
Collision Reward	1.0×10^3
TTC Reward	1.0×10^2
Speed Reward	1.0×10^1
Lane Change Reward	1.0×10^0
Merging Task Reward	1.0×10^0

Table 4.2: Parameters of Experiment

Parameters	Value
Simulator	Highway-env
Total Length	740
Entrance of Merge	320
Merge	200
Exit of Merge	220
Position Noise	[-1.5, 1.5]
Initial Speed of Main-lane	25 - 27 m/s
Initial Speed of Merging-lane	20 - 22 m/s
Action Frequency	1 seconds
Number of CAV in First Main-lane	1
Number of CAV in Second Main-lane	1
Number of CAV in Merging-lane	1
Number of HDV in First Main-lane	2
Number of HDV in Second Main-lane	2
Number of HDV in Merging-lane	1
Total Training Episodes	15000
Optimizer	RMSprop
Gradient Clipping	10
γ	0.99
Target Update Cycle	200
Train Seed	Starting from 2024
Test Cycle	100 Episodes
Test Seed	0, 25, 50, …, 1000
w_c	1000
w_s	1
w_{ttc}	1
w_l	1
w_m	10

4.1.2 Vehicle Kinematic Model

In `highway-env`, the Kinematic Bicycle Model (Polack et al. 2017) governs the kinematics of vehicles, as depicted in Equation 4.1. This model mathematically represents the vehicle's motion using the following state equations:

$$\begin{aligned}
\dot{x} &= v\cos(\psi + \beta) && \text{(rate of change of x-position)} \\
\dot{y} &= v\sin(\psi + \beta) && \text{(rate of change of y-position)} \\
\dot{v} &= a && \text{(acceleration)} \\
\dot{\psi} &= \frac{v}{l_r}\sin\beta && \text{(rate of change in heading)} \\
\beta &= \tan^{-1}\left(\frac{l_r}{l_r + l_f}\tan\delta\right) && \text{(slip angle at the center of gravity)}
\end{aligned} \tag{4.1}$$

In this model, the vehicle's position is represented by coordinates (x, y), where x and y are the longitudinal and lateral positions, respectively. The term v denotes the vehicle's forward speed, while ψ indicates the heading angle. The acceleration of the vehicle, denoted by a, is an input to the model, reflecting the acceleration command. The slip angle at the vehicle's center of gravity, represented by β, and the steering angle of the front wheels, denoted by δ, influence the trajectory and orientation of the vehicle. l_f and l_r represent the distances from the vehicle's center of gravity to the front and rear axles, respectively. The model assumes a simple relationship between the front wheel steering angle and the slip angle, capturing the essence of the vehicle's dynamic response to steering inputs in a straightforward manner.

4.1.3 Vehicle Low-level Controller

This section describes the low-level control mechanisms implemented in the `highway-env` simulator, which our project utilizes without modification. The simulator employs a proportional–integral–derivative (PID) controller to convert high-level meta-actions into specific steering and throttle control signals. We have maintained most of the simulator's default parameter values for this controller, However, we made one adjustment to accommodate low speed limit: we expanded the desired speed list by adding two speeds [10, 15] to the existing default values, maintaining the same interval 5 as the original setting.

- ***Longitudinal Controller:***

  $$a = K_p(v_r - v) \tag{4.2}$$

  In Equation 4.2, a represents the vehicle's acceleration, which can be controlled through throttle modulation. Here, v is the current vehicle velocity, and v_r is target velocity, which is retrieved from a list of desired speeds. The proportional gain, K_p, determines the response intensity of the controller to the velocity error, thus regulating how aggressively the vehicle accelerates or decelerates to match the desired speed.

- ***Lateral Controller:***

  $$\begin{aligned}
  v_{lat,r} &= -K_{p,lat}\Delta_{lat} \\
  \Delta\psi_r &= \arcsin\left(\frac{v_{lat,r}}{v}\right)
  \end{aligned} \tag{4.3}$$

  $$\begin{aligned}
  \psi_r &= \psi_L + \Delta\psi_r \\
  \dot{\psi}_r &= K_{p,\psi}(\psi_r - \psi) \\
  \delta &= \arcsin\left(\frac{1}{2}\frac{l}{v}\dot{\psi}_r\right)
  \end{aligned} \tag{4.4}$$

  In Equation 4.3, Δ_{lat} quantifies the lateral deviation of the vehicle from the centerline of the lane, and $v_{lat,r}$ is the resultant lateral velocity command calculated to correct this deviation. The term $\Delta\psi_r$ represents the required change in heading to achieve $v_{lat,r}$, effectively aligning the vehicle with the lane.

  In Equation 4.4, ψ_L denotes the heading of the lane. The controller computes ψ_r, the target heading, which integrates the lane heading with the necessary adjustments from $\Delta\psi_r$. The yaw rate command, $\dot{\psi}_r$, and the front wheel steering angle, δ, are then derived to ensure the vehicle follows this calculated trajectory. Control gains $K_{p,lat}$ and $K_{p,\psi}$ adjust the sensitivity of the lateral position and heading control responses, respectively.

4.1.4 Behaviors of Human Vehicles

In `highway-env`, HDVs use the Intelligent Driver Model (IDM) (Treiber, Hennecke, and Helbing 2000) for longitudinal behavior and the MOBIL model (Kesting, Treiber, and Helbing 2007) for lateral behavior, including lane changes. Our project retains the default parameter settings for both models.

- ***Longitudinal Behavior:***

$$\begin{aligned} \dot{v} &= a\left[1-\left(\frac{v}{v_0}\right)^{\delta}-\left(\frac{d^*}{d}\right)^{2}\right] \\ d^* &= d_0 + Tv + \frac{v\Delta v}{2\sqrt{ab}} \end{aligned} \tag{4.5}$$

  In this model, v represents the vehicle's current velocity, and d is the distance to the vehicle ahead. The parameters are defined as follows: v_0 is the desired velocity, T is the desired time headway, d_0 is the minimum jam distance, and a and b are the maximum acceleration and comfortable deceleration parameters, respectively. The velocity exponent δ influences the sensitivity of the acceleration to speed.

- ***Lateral Behavior:***

$$\tilde{a}_n \geq -b_{\text{safe}} \tag{4.6}$$

  This safety condition ensures that the new follower's acceleration $\tilde{a}_n$ does not necessitate braking more than b_{safe}, which is the safe braking limit.

$$\underbrace{\tilde{a}_c - a_c}_{\text{ego-vehicle}} + p\left(\underbrace{\tilde{a}_n - a_n}_{\text{new follower}} + \underbrace{\tilde{a}_o - a_o}_{\text{old follower}}\right) \geq \Delta a_{\mathrm{th}} \tag{4.7}$$

  In this incentive condition, c denotes the ego vehicle, o is its old follower prior to the lane change, and n is the new follower post-lane change. The accelerations a and $\tilde{a}$ are those before and after the lane change, respectively. The term p represents a politeness factor which weighs the benefits to other vehicles, and Δa_{th} is the minimum net acceleration gain required to justify a lane change.

4.2 Training Performance

This study presents a comprehensive evaluation of our proposed QMIX-QLambdaM algorithm for CAVs on-ramp merging. Our experimental approach consists of two main components: a detailed sensitivity analysis and a comparative study against state-of-the-art benchmarks. The sensitivity analysis focuses on three key aspects of our algorithm: the impact of different λ values in QMIX-QLambdaM; the effect of varying the number of surrounding vehicles considered in TTC evaluation; and the influence of the action masking mechanism on training efficiency and safety performance. Following this, we conduct a thorough comparison of QMIX-QLambdaM against three state-of-the-art algorithms: QMIX, MAA2C, and COMA, evaluating performance across multiple metrics including reward accumulation, collision rates, and average speeds.

4.2.1 Sensitivity analysis

4.2.1.1 Comparisons of Different λ in QMIX-QLambdaM

The choice of λ in QMIX-QLambdaM plays a crucial role in balancing between TD learning and MC approaches. Values of λ closer to 0 bias the learning strategy towards TD learning, while values nearing 1 align more closely with the MC approach. To identify the optimal value of λ for effective learning, we conducted a sensitivity analysis using a discrete set of λ values: 0.3, 0.5, 0.6, and 0.9. This granularity is inspired by (J. Hu et al. 2023). While a finer granularity would provide more precise results, computational resources, and time constraints limited our analysis to these specific values.

Figure 4.2 illustrates the moving average reward learning curves for different λ values. The graph demonstrates that $\lambda = 0.6$ consistently outperforms other values, achieving higher rewards throughout the training process. This superior performance is particularly evident in the later stages of training after 7000 episodes.

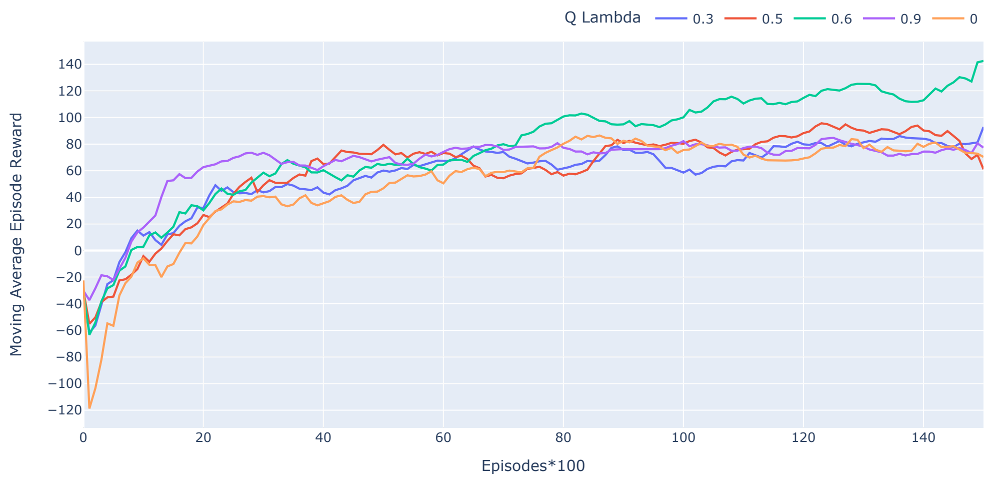


Figure 4.2: Moving Average Reward Learning Curve of Different Lambda (n=9 for better trend visibility): $\lambda = 0.6$ comes out since 7000 episodes collected

Table 4.3 provides a comprehensive comparison of performance metrics across different λ values. For the collision indicator, $\lambda = 0.9$ has the lowest mean value, $\lambda = 0.6$ is the second best performer, matching the performance of $\lambda = 0.3$ while maintaining higher rewards. Regarding average speed, $\lambda = 0$ (original QMIX) achieves the highest mean speed. Followed by $\lambda = 0.6$, which is the second best performer. This indicates that $\lambda = 0.6$ allows for more efficient maneuvers without compromising safety. In particular, in terms of rewards, $\lambda = 0.6$ significantly outperforms other values.

The comparative analysis presented in Figure 4.3, Figure 4.4, and Figure 4.5 provides compelling evidence for selecting $\lambda = 0.6$ through comprehensive distribution analysis of key performance indicators. Specifically, Figure 4.3 demonstrates that $\lambda = 0.6$ not only achieves the highest median reward but also exhibits a broader distribution in the upper reward range, suggesting enhanced capability in maximizing advantageous scenarios. The collision analysis in Figure 4.4 reveals that $\lambda = 0.6$ maintains consistently low collision rates, comparable to other λ values with the exception of $\lambda = 0.9$. Meanwhile, Figure 4.5 validates its exceptional performance in maintaining high average speeds, second only to the original QMIX implementation. Collectively, these boxplot analyses substantiate that $\lambda = 0.6$ achieves the optimal balance between safety and efficiency while delivering superior overall reward performance.

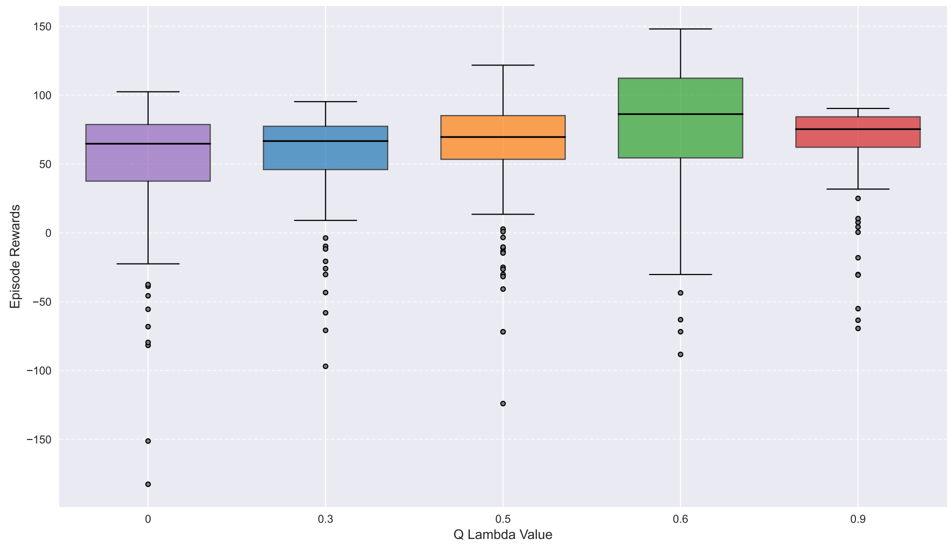


Figure 4.3: Statistics of Reward of Different Lambda of 150 Test Phases Over the Period of Training

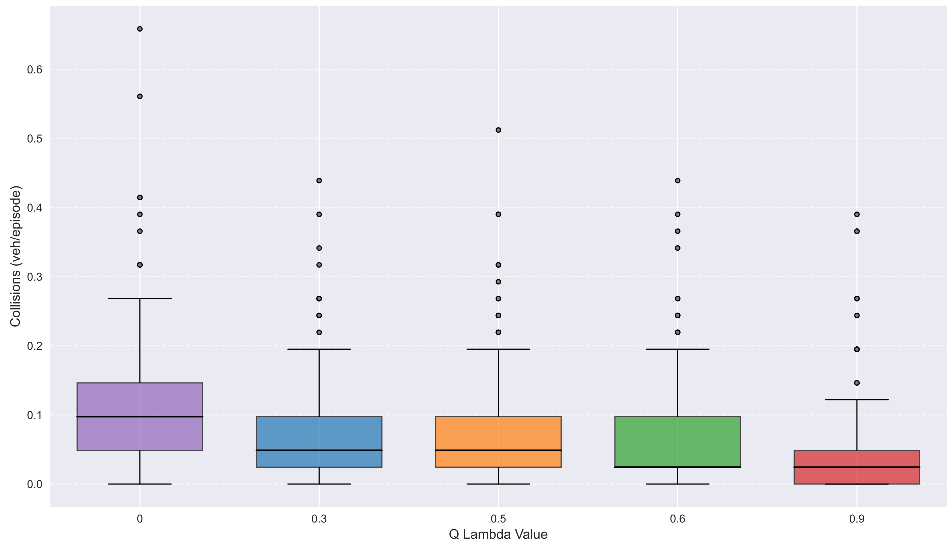


Figure 4.4: Statistics of Collision of Different Lambda of 150 Test Phases Over the Period of Training

Table 4.3: Comprehensive Comparison of Speed, Collision, and Reward across Different Lambda

Indicator	Statistic	0	0.3	0.5	0.6	0.9
Collision (veh/episode)	Mean ± std	0.12 ± 0.10	0.07 ± 0.08	0.08 ± 0.09	0.07 ± 0.08	**0.05 ± 0.07**
Speed (m/s)	Mean ± std	**22.4 ± 0.8**	20.8 ± 0.7	21.3 ± 1.3	21.6 ± 1.2	18.6 ± 0.8
Reward	Mean ± std	51 ± 44	57 ± 33	61 ± 39	**78 ± 45**	66 ± 29

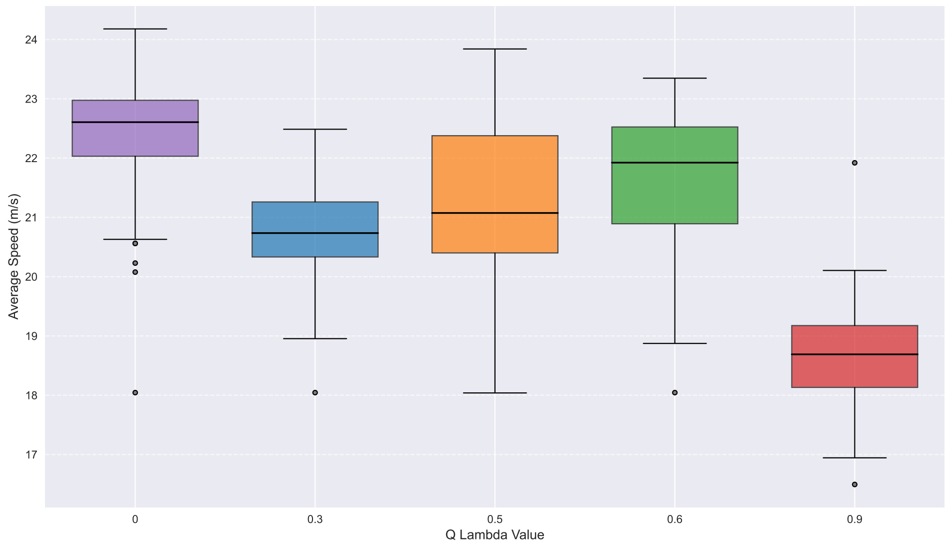


Figure 4.5: Statistics of Average Speed of Different Lambda of 150 Test Phases Over the Period of Training

4.2.1.2 Comparison of Different Surrounding Vehicles in TTC Evaluation

The selection of an appropriate number of surrounding vehicles (n_{surr}) for TTC evaluation is crucial for optimal performance. As outlined in the methodology chapter, an ablation study was conducted to determine the most effective configuration. Given that the observable vehicle count ranges from 1 to 6, our sensitivity analysis encompasses this full range. It's important to note that scenarios with a greater number of vehicles tend to yield higher TTC reward values, as the rewards for larger TTC values offset those for smaller ones. To ensure a fair comparison, we focus on collision rates and average speeds rather than TTC rewards or total rewards.

Analysis of the learning curves and statistical data reveals that utilizing 6 surrounding vehicles ($n_{surr} = 6$) yields the most favorable outcomes. Figure 4.6 illustrates the learning curves for average speeds, where the 6-vehicle scenario consistently maintains high speeds after 6500 episodes are collected in the training period. Concurrently, Figure 4.7 demonstrates that this configuration achieves a rapid decrease in collisions early in training and maintains one of the lowest collision rates across episodes. These learning curves indicate superior performance in both speed and safety domains.

Further supporting this conclusion, Figure 4.8 shows that the 6-vehicle setup achieves one of the highest median speeds and the distributions are higher than other configurations. Similarly, Figure 4.9 reveals that this configuration attains one of the lowest collision distributions, highlighting its strong safety performance. The comprehensive comparison in Table 4.4 corroborates these findings, showing that the 6-vehicle setup strikes an optimal balance between speed and safety.

Table 4.4: Comprehensive Comparison of Speed, Collision Across Different Evaluated Vehicles

Indicator	Statistic	n_{surr} = 1	n_{surr} = 2	n_{surr} = 3	n_{surr} = 4	$n_{surr} = 5$	$n_{surr} = 6$
Collision (cav/episode)	Mean $\pm$ std	0.07 $\pm$ 0.08	**0.06** $\pm$ **0.08**	0.08 $\pm$ 0.08	0.08 $\pm$ 0.09	0.07 $\pm$ 0.08	**0.06** $\pm$ **0.09**
Speed (m/s)	Mean $\pm$ std	**21.6** $\pm$ **1.2**	21.1 $\pm$ 1.4	20.8 $\pm$ 1.8	21.3 $\pm$ 1.6	21.5 $\pm$ 1.3	**21.6** $\pm$ **2.0**

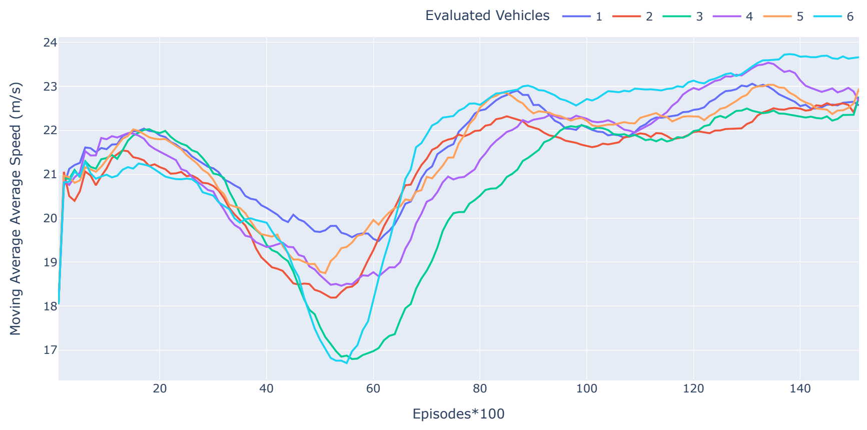


Figure 4.6: Moving Average Speed Learning Curve of Different Evaluated Vehicles (n=9 for better trend visibility): 6-vehicle setup dominate since 6500 episodes are collected

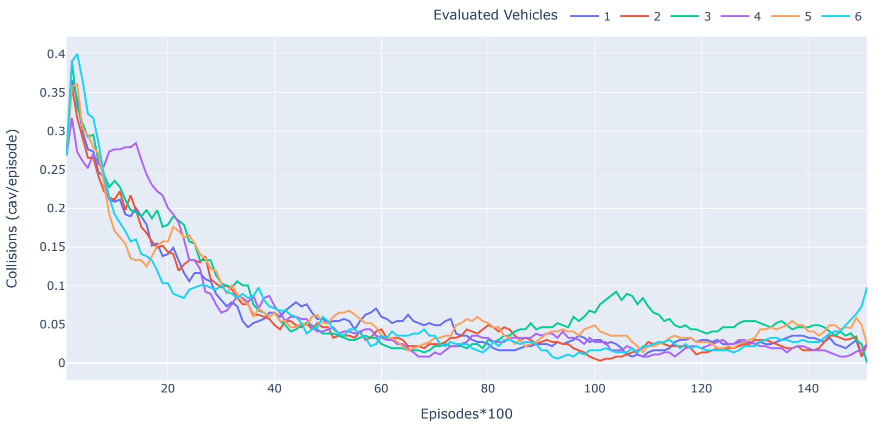


Figure 4.7: Moving Average Collision Learning Curve of Different Evaluated Vehicles (n=9 for better trend visibility): 6-vehicle setup achieves a rapid decrease in collisions early in training and maintains one of the lowest collision rates across episodes and converges faster.

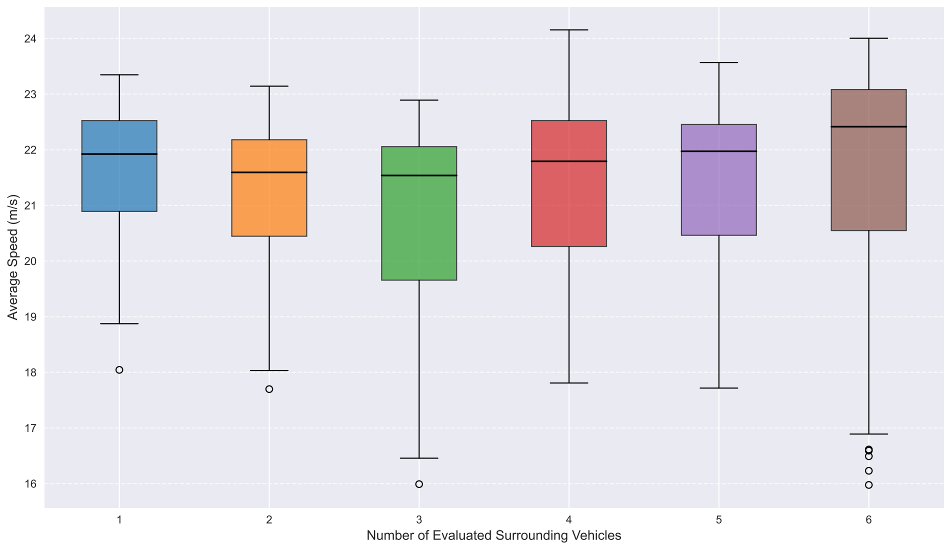


Figure 4.8: Statistics of Speed of Different Evaluated Vehicles of 150 Test Phases Over the Period of Training

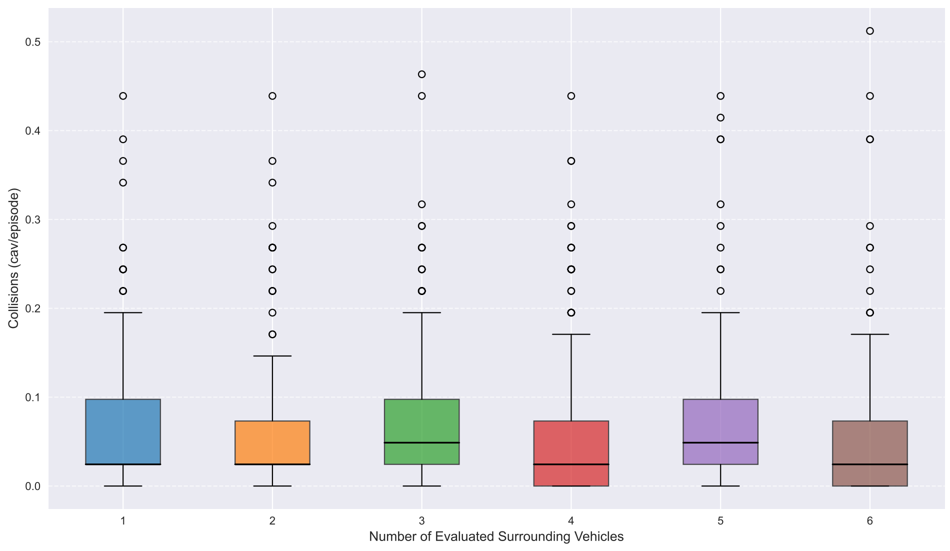


Figure 4.9: Statistics of Collision of Different Evaluated Vehicles of 150 Test Phases Over the Period of Training

4.2.1.3 The Impact of Action Masking Mechanism of Training

As outlined in chapter 3, we developed an action masking mechanism aimed at accelerating training speed and enhancing safety performance. This section presents a comprehensive analysis of the mechanism's impact by comparing policies with and without action masking (original QMIX). Our evaluation focuses on two key aspects: training efficiency, assessed through rewards learning curves, and safety performance, measured by collision statistics from both the first test phase (first guess) and across 150 test phases.

Figure 4.10 presents the moving average episode reward under both conditions. The policy with action masking demonstrates superior performance throughout the training process, particularly notable in its higher initial reward values. This pronounced advantage in the early stages manifests as an enhanced learning curve, confirming that action masking successfully accelerates the learning process.

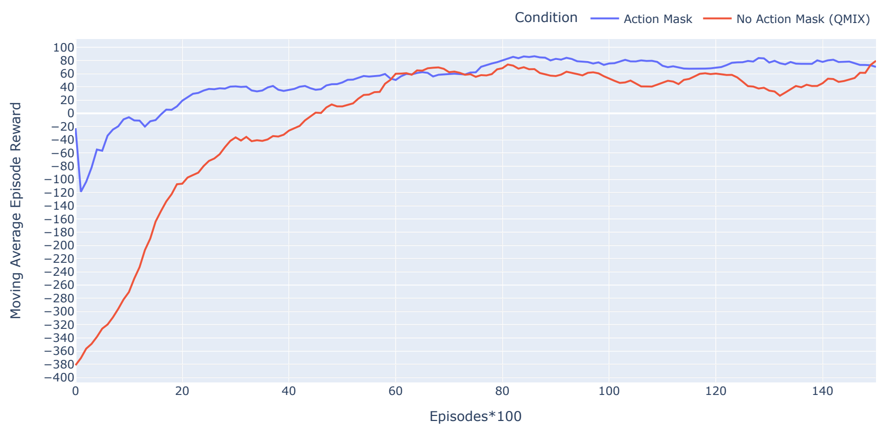


Figure 4.10: Moving Average Reward Learning Curve of Two Conditions (n=9 for better trend visibility): action mask setup shows higher initial reward

Regarding safety performance, Figure 4.11, Table 4.5 collectively highlight significant improvements. The action mask policy exhibits markedly enhanced safety metrics in both the first guess and across the 150 test phases, with substantially lower mean collision rates compared to the baseline policy. Notably, the first guess scenario shows remarkable improvement: prior to implementing action masking, the collision rate was 1.27 vehicles per episode, indicating at least one CAV collision per scenario. The introduction of action masking led to a substantial 70.9% reduction in mean collision rates. Furthermore, Figure 4.11 demonstrates that the action mask policy yields a more compact and reduced distribution of collision rates across both the 150 test phases and the initial test phase. This pattern not only highlights the policy's remarkable collision-avoidance capabilities even in its untrained state but also underscores its enhanced consistency and reliability in maintaining safety standards across diverse scenarios throughout the training process.

Table 4.5: Comprehensive Comparison of Collision of Two Conditions

Indicator	Statistic	Action Mask	No Action Mask (QMIX)
Collision of the First Test Phase (cav/episode)	Mean $\pm$ std	**0.37 $\pm$ 0.65**	1.27 $\pm$ 0.54
Collision of 150 Test Phases (cav/episode)	Mean $\pm$ std	**0.12 $\pm$ 0.10**	0.26 $\pm$ 0.28

In conclusion, the action masking mechanism substantially improves both training efficiency and safety performance. The faster learning rate and higher overall rewards indicate that action masking effectively guides the policy towards optimal behaviors, potentially reducing the required training time. The significant reduction in mean and maximum collision rates, coupled with lower variability, demonstrates that action masking enhances safety factors considerably. These findings strongly support the initial goals of expediting training speed and improving safety through the implementation of action masking.

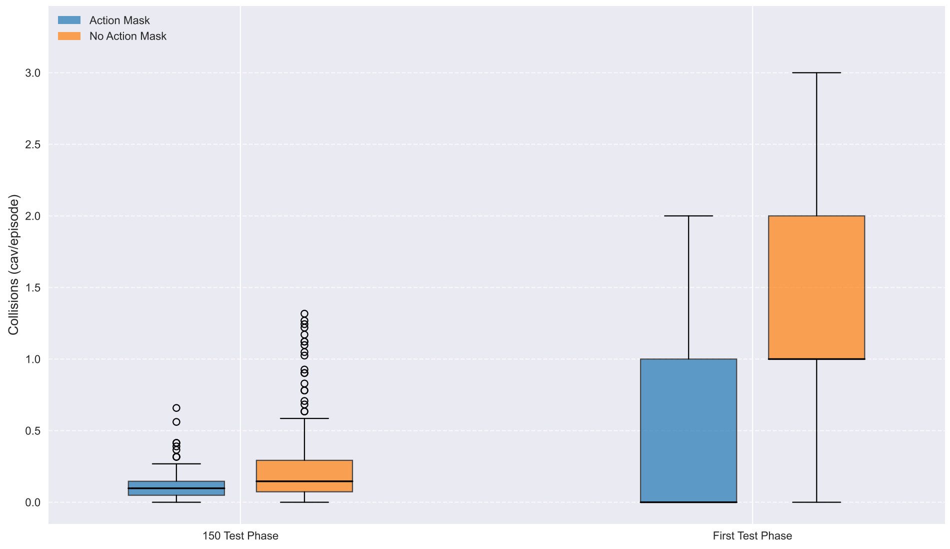


Figure 4.11: Statistics of Collision of Two Conditions

4.2.2 Benchmark Comparison

This section compares our proposed algorithm, QMIX-QLambdaM, with three state-of-the-art (SOTA) baselines: QMIX, MAA2C, and COMA. MAA2C extends the A2C (Advantage Actor-Critic) algorithm to multi-agent settings by introducing a centralized critic to evaluate joint rewards, enabling simultaneous training of agents using decentralized actors. COMA builds upon MAA2C by addressing credit assignment through a counterfactual baseline that marginalizes out individual agent actions while keeping others fixed. The learning parameters for QMIX and QMIX-QLambdaM in Table 4.6 are adapted from (Rashid, Samvelyan, C. S. d. Witt, et al. 2018) with some coarse tuning, while MAA2C and COMA parameters in Table 4.7 are based on (Foerster et al. 2017). For QMIX-QLambdaM, we set λ to 0.6, determined through sensitivity analysis detailed in subsection 4.2.1.

Table 4.6: The learning parameters of QMIX and QMIX-QLambdaM which are borrowed from (Rashid, Samvelyan, C. S. d. Witt, et al. 2018), λ is only used for QMIX-QLambdaM

Parameters	QMIX-QLambdaM & QMIX
FC_1	[15(7+5+3), 64]
GRU	[64, 64]
FC_2	[64, 5]
FC_W1	[45(5*15), 96(3*32)]
FC_b1	[45, 32]
FC_W2	[45, 32]
FC_b21	[45, 32]
FC_b22	[32, 1]
Learning Rate	0.0005
Initial Value of ϵ	1.0
Final ϵ Value	0.05
Annealing episodes for ϵ	1100
Batch Size	128
Replay Buffer Size	5000
λ in Q (λ)	0.6

A comprehensive analysis of the results demonstrates several notable advantages of QMIX-QLambdaM compared to the baseline algorithms. As illustrated in Figure 4.12 and Figure 4.13, QMIX-QLambdaM exhibits superior performance in terms of rewards, maintaining the highest reward learning curve throughout the training process and achieving the highest position in the boxplot distribution. This superiority is quantitatively validated in Table 4.8, where QMIX-QLambdaM attains the highest mean reward, surpassing the next best performer, QMIX, by a significant margin of 91.

Table 4.7: The learning parameters of MAA2C and COMA which are borrowed from (Foerster et al. 2017)

Parameters	MAA2C	COMA
Actor_FC_1	[15, 64]	[15, 64]
Actor_GRU	[64, 64]	[64, 64]
Actor_FC_2	[64, 5]	[64, 5]
Critic_FC_1	[45, 128]	[85(45+7+3+5*3*2), 128]
Critic_FC_2	[128, 128]	[128, 128]
Critic_FC_3	[128, 1]	[128, 5]
Learning rate of actor	0.0005	0.0005
Learning rate of critic	0.0005	0.0005
Initial Value of ϵ	0.5	0.5
Final ϵ Value	0.02	0.02
Annealing episodes for ϵ	750	750
λ in TD (λ)	/	0.8

Regarding safety performance, Table 4.8 indicates that QMIX-QLambdaM achieves the lowest collision rate among all algorithms, markedly outperforming the second-best algorithm, QMIX, with a substantial improvement of 74.1%. The collision distribution of the proposed method also demonstrates the most compact and favorable profile.

In terms of efficiency, Figure 4.15 and Table 4.8 reveal that QMIX-QLambdaM maintains the second-highest average speed, showing only a marginal decrease of 5.0% compared to QMIX, while slightly outperforming MAA2C by 3.0% and significantly exceeding COMA by 33.3%. These results suggest that while QMIX-QLambdaM trades off a modest amount of efficiency to achieve excellent collision avoidance capabilities, the compromise in performance is minimal.

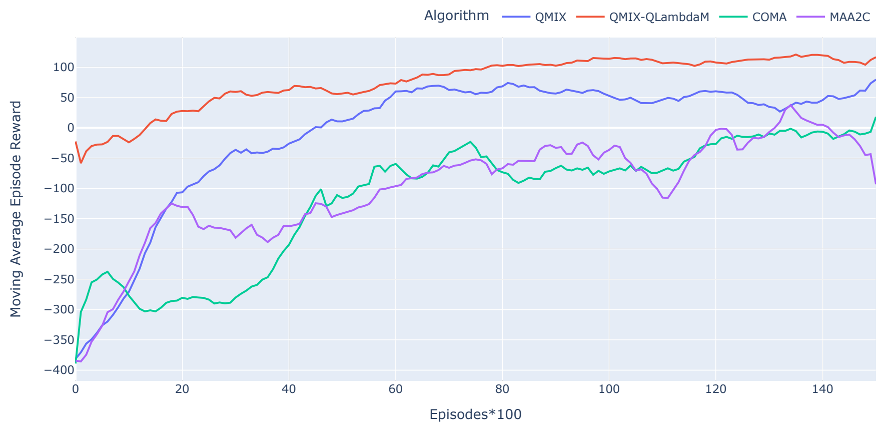


Figure 4.12: Moving Average Reward Learning Curve of Different Algorithms (n=9 for better trend visibility)

Table 4.8: Comprehensive Comparison of Speed, Collision across Different Algorithms

Indicator	Statistic	QMIX-QLambdaM	QMIX	COMA	MAA2C
Collision (cav/episode)	Mean ± std	**0.07 ± 0.09**	0.27 ± 0.31 (-74.1%)	0.37 ± 0.23 (-81.1%)	0.35 ± 0.24 (-80.0%)
Speed (m/s)	Mean ± std	20.7 ± 1.9	**21.8 ± 1.3** (-5.0%)	14.5 ± 2.7 (+42.8%)	20.1 ± 4.5 (+3.0%)
Reward	Mean ± std	**86 ± 53**	-5 ± 124 (+91)	-55 ± 90 (+141)	-32 ± 106 (+118)

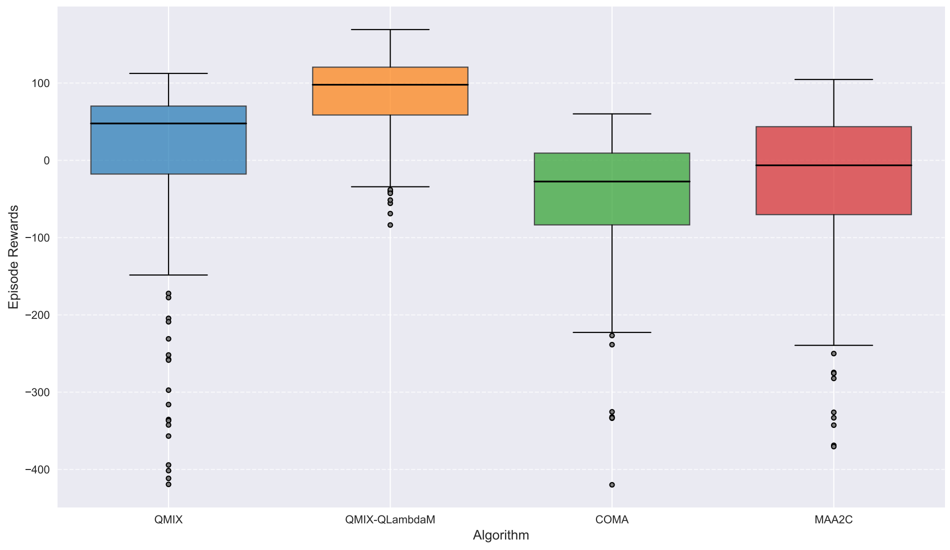


Figure 4.13: Statistics of Reward of Different Algorithms of 150 Test Phases Over the Period of Training

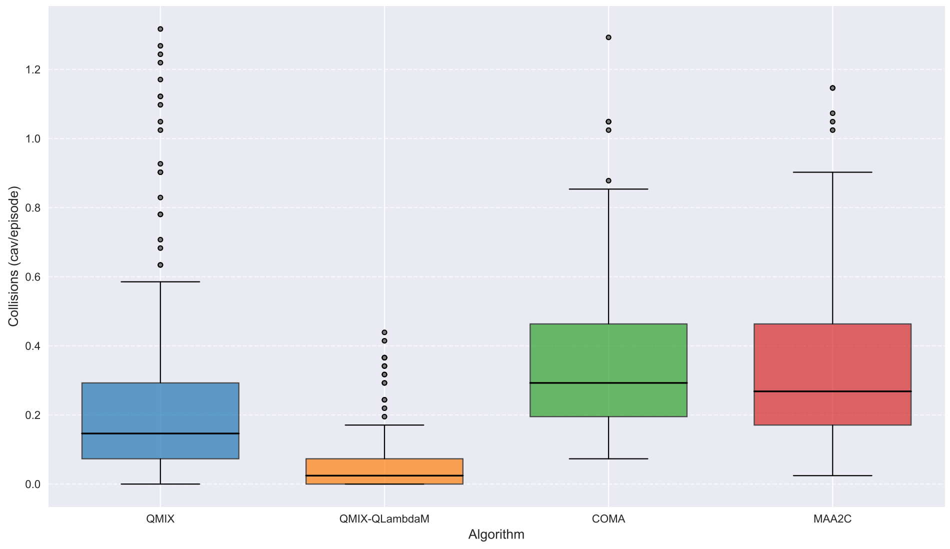


Figure 4.14: Statistics of Collision of Different Algorithms of 150 Test Phases Over the Period of Training

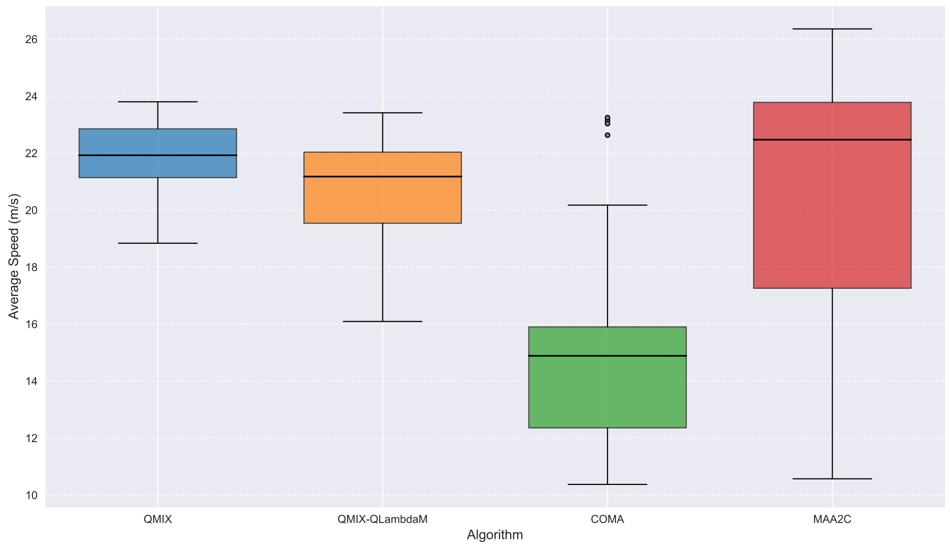


Figure 4.15: Statistics of Average Speed of Different Algorithms of 150 Test Phases Over the Period of Training

4.3 Evaluations & Results

This section presents a comprehensive evaluation of our proposed QMIX-QLambdaM algorithm's performance in adaptive on-ramp scenarios, consisting of two main components: a performance analysis under varying traffic densities and an in-depth examination of strategic control for main-lane and merging-lane vehicles. In the first part, we assess the algorithm's adaptability and robustness by testing it under low, medium, and high traffic density conditions, comparing QMIX-QLambdaM's performance against state-of-the-art baselines (QMIX, COMA, and MAA2C) across multiple metrics, including total reward, safety reward, efficiency reward, merging task reward, collision rates, and average speeds. The second part focuses on strategic control capabilities, showcasing QMIX-QLambdaM's effectiveness in handling complex traffic situations for both main-lane and merging-lane vehicles through detailed case studies comparing our algorithm with the best-performing baseline (MAA2C) in challenging scenarios.

4.3.1 Evaluations in Different Traffic Density

To test the trained policy and evaluate the adaptive ability of our proposed method, we employed the trained policy for cases with different vehicle densities, changing the number of HDVs for each lane. The densities are categorized into three levels: low density which is 9.09 veh/km (3 HDVs, each lane has one HDV), medium density, which is 13.64veh/km (6 HDVs, each lane has two HDVs) and high density, which is 18.18veh/km (9 HDVs, each lane has three HDVs). According to the flow-density diagram in (Treiber 2013), traffic can exist in either free or congested states within a certain density range (18-25 veh/km), with the upper bound of this range still corresponding to free flow conditions. We selected a policy with the best performance (highest reward value) for each algorithm and compared them across these three density levels. The experiments were run with 40 different seeds, as described in section 2.3. The evaluation metrics include total reward, safety reward, efficiency reward, merging task reward, collision rate, and average speed.

The experimental results across varying traffic densities demonstrate distinct performance patterns for each algorithm, as illustrated in Figure 4.16–Figure 4.21 and Table 4.9. QMIX-QLambdaM demonstrates consistently superior performance, achieving not only the highest mean total rewards but also displaying notably elevated quartile distributions compared to other algorithms, as evidenced by its distinctively higher boxplot position. In low traffic density scenarios, QMIX-QLambdaM's mean total reward surpasses QMIX, COMA, and MAA2C by margins of 84, 130, and 115 respectively. This performance advantage persists in medium density conditions, with margins of 77, 79, and 93 respectively. Similarly, in high-density scenarios, QMIX-QLambdaM maintains its lead with advantages of 109, 123, and 104 respectively.

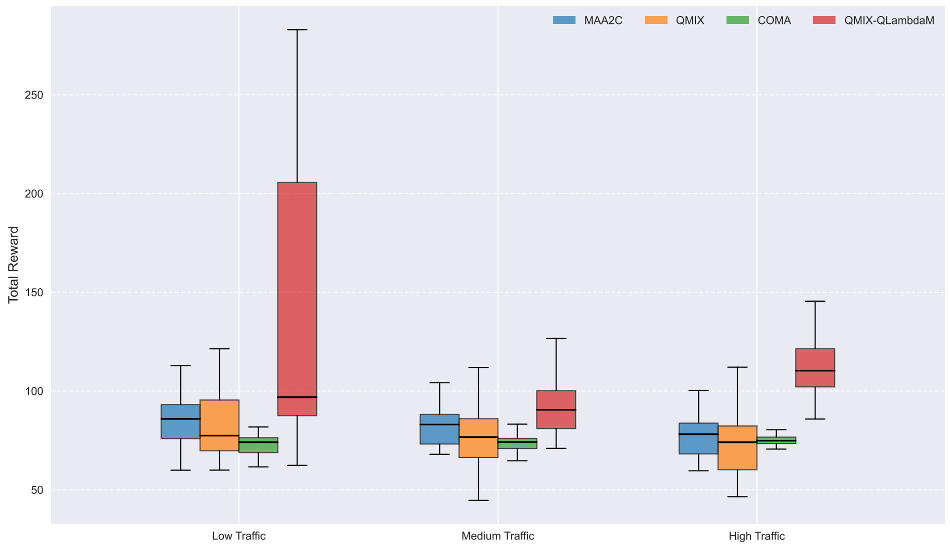


Figure 4.16: Statistics of Total Reward for Different Traffic Density

Safety performance, evaluated through collision rates and safety rewards (comprising collision reward and TTC reward), reveals notable distinctions among the algorithms. QMIX-QLambdaM exhibits exceptional safety characteristics, maintaining zero collision rates and high safety rewards across all density levels, demonstrating its ability to maintain safe gaps and prevent collisions in diverse scenarios which are shown in Figure 4.17 and Figure 4.18. In contrast, other algorithms exhibit non-zero collision rates across all traffic densities, moreover, Figure 4.18 shows that each of the baseline even has two CAVs colliding in the experiments.

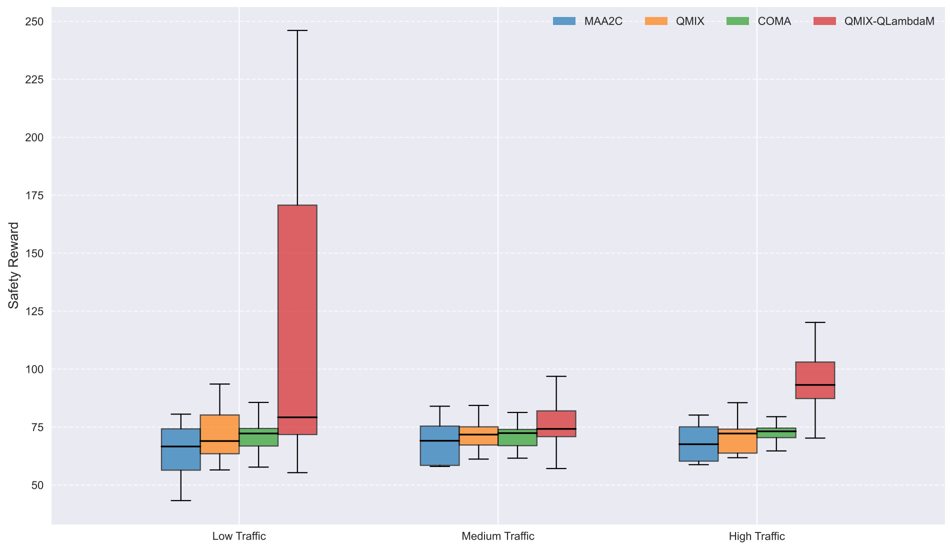


Figure 4.17: Statistics of Safety Reward for Different Traffic Density

Efficiency performance is evaluated through average speed and efficiency rewards (comprising speed reward and lane change reward) as shown in Figure 4.20, Figure 4.19 and Table 4.9. The average speed data reveals QMIX-QLambdaM's superior efficiency across all densities. In low density conditions, the proposed method outperforms QMIX, COMA, and MAA2C by 10.8%, 80.3%, and 16% respectively. These advantages persist in medium density (7.6%, 58.2%, and 7.1%) and heavy density scenarios (11.3%, 54.5%, and 11.9%). The efficiency rewards further validate this superior speed performance, with QMIX-QLambdaM achieving the highest values across all densities. Notably, while

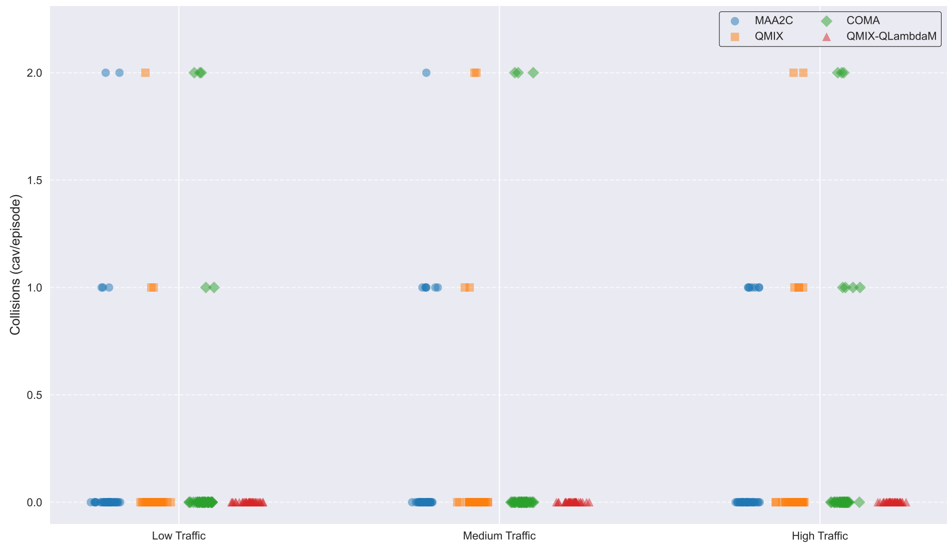


Figure 4.18: Statistics of Collisions for Different Traffic Density: the figure is shown with scatter as the boxplot is not readable.

MAA2C demonstrates a clear advantage in efficiency rewards compared to QMIX despite similar average speeds, this disparity stems from QMIX's excessive lane changes.

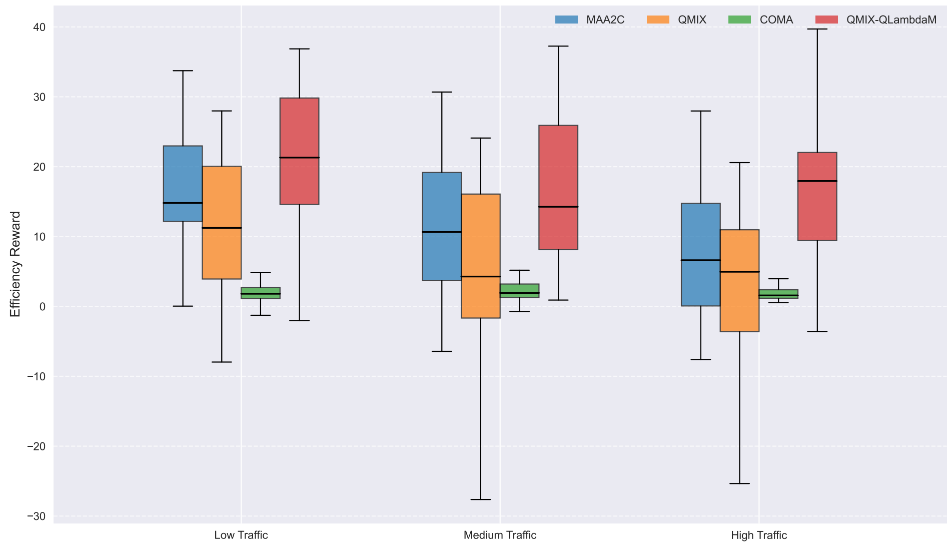


Figure 4.19: Statistics of Efficiency Reward for Different Traffic Density

Merging task rewards, which indicate merging efficiency, exhibit varying performance patterns across algorithms and densities. MAA2C achieves optimal mean rewards in low density conditions, while QMIX-QLambdaM excels in medium density scenarios, and COMA demonstrates superior performance in high density situations. Overall, QMIX-QLambdaM demonstrates better merging task rewards than QMIX while performing marginally below the actor-critic methods. However, the high collision rates exhibited by the actor-critic method during merging suggests a trade-off where safety is compromised to achieve merging objectives.

The standard deviations in total rewards provide valuable insights into the consistency of each algorithm's performance. QMIX-QLambdaM exhibits notably lower standard deviations compared to other algorithms, indicating more consistent performance across diverse traffic scenarios. Furthermore, QMIX-QLambdaM demonstrates exceptional stability, particularly in high-density scenarios, where it maintains both superior performance and lower standard deviations in total rewards compared to al-

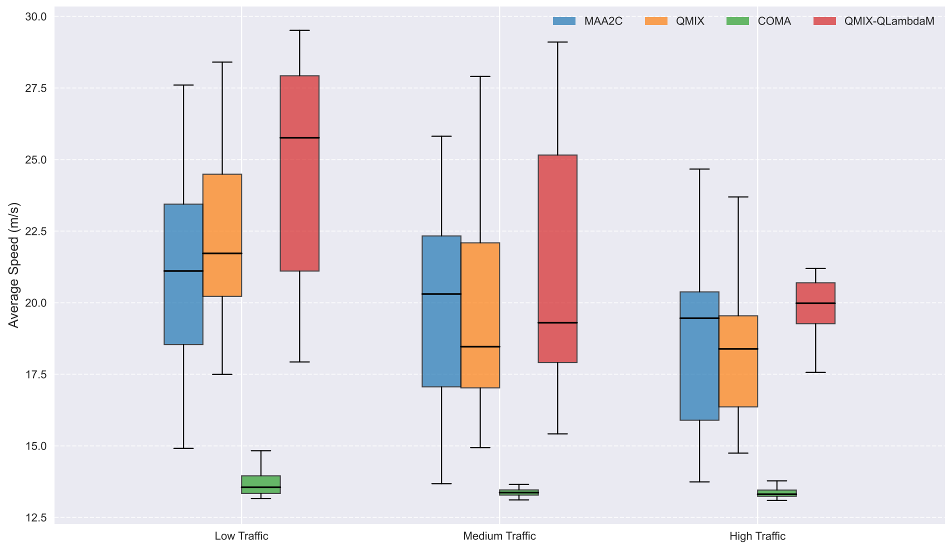


Figure 4.20: Statistics of Average Speed for Different Traffic Density

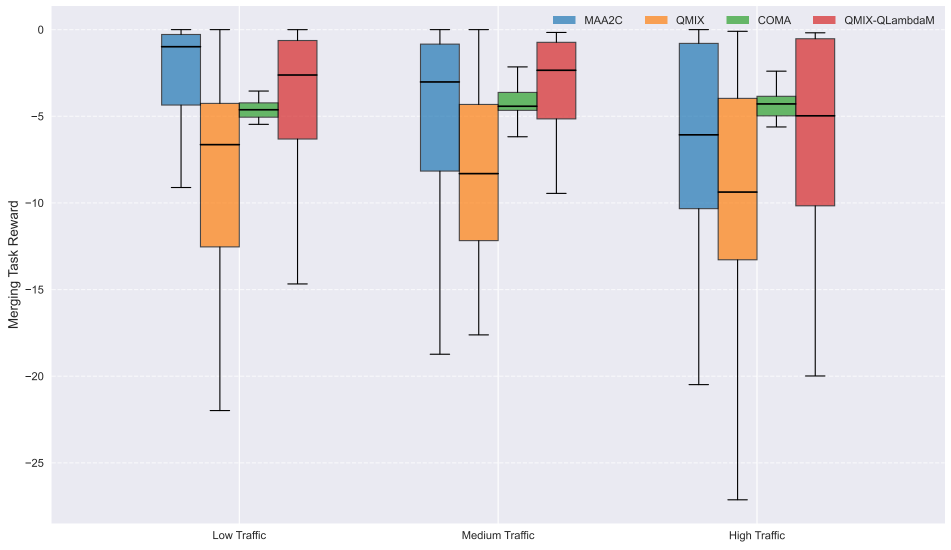


Figure 4.21: Statistics of Merging Task Reward for Different Traffic Density

ternative methods.

Table 4.9: Comprehensive Comparison of Total Reward, Safety Reward, Efficiency Reward, Merging Task Reward, Collision, and Average Speed across Different Algorithms in Different Traffic Densities: the proposed method outperforms nearly in every indicator except the merging task reward in low and heavy density

Traffic Density	Indicator	Statistic	QMIX-QLambdaM	QMIX	COMA	MAA2C
Low	Total Reward	Mean ± std	**137 ± 69**	53 ± 144 (+84)	7 ± 203 (+130)	22 ± 184 (+115)
	Safety Reward	Mean ± std	**115 ± 62**	41 ± 139 (+74)	5 ± 201 (+110)	5 ± 179 (+110)
	Efficiency Reward	Mean ± std	**22 ± 11**	12 ± 10 (+10)	2 ± 2 (+20)	17 ± 9 (+5)
	Merging Task Reward	Mean ± std	-5 ± 6	-8 ± 6 (+3)	-5 ± 1 (0)	**-3 ± 5** (-2)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.10 ± 0.37 (-100%)	0.20 ± 0.55 (-100%)	0.17 ± 0.49 (-100%)
	Average Speed (m/s)	Mean ± std	**24.7 ± 3.7**	22.3 ± 2.9 (+10.8%)	13.7 ± 0.4 (+80.3%)	21.3 ± 3.4 (+16.0%)
Medium	Total Reward	Mean ± std	**104 ± 43**	27 ± 172 (+77)	25 ± 186 (+79)	11 ± 169 (+93)
	Safety Reward	Mean ± std	**88 ± 37**	21 ± 170 (+67)	23 ± 185 (+65)	-1 ± 165 (+89)
	Efficiency Reward	Mean ± std	**17 ± 10**	5 ± 12 (+12)	2 ± 2 (+15)	12 ± 10 (+5)
	Merging Task Reward	Mean ± std	**-3 ± 3**	-9 ± 7 (+6)	-4 ± 1 (+1)	-5 ± 6 (+2)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.15 ± 0.47 (-100%)	0.15 ± 0.52 (-100%)	0.20 ± 0.45 (-100%)
	Average Speed (m/s)	Mean ± std	**21.2 ± 4.2**	19.7 ± 3.7 (+7.6%)	13.4 ± 0.2 (+58.2%)	19.8 ± 3.2 (+7.1%)
High	Total Reward	Mean ± std	**116 ± 30**	7 ± 177 (+109)	-7 ± 208 (+123)	12 ± 149 (+104)
	Safety Reward	Mean ± std	**100 ± 26**	4 ± 176 (+96)	-9 ± 207 (+109)	5 ± 144 (+95)
	Efficiency Reward	Mean ± std	**17 ± 10**	3 ± 11 (+14)	2 ± 2 (+15)	7 ± 9 (+10)
	Merging Task Reward	Mean ± std	-6 ± 6	-10 ± 8 (+4)	**-4 ± 1** (-2)	-7 ± 6 (+1)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.20 ± 0.50 (-100%)	0.24 ± 0.58 (-100%)	0.17 ± 0.38 (-100%)
	Average Speed (m/s)	Mean ± std	**20.7 ± 2.6**	18.6 ± 2.9 (+11.3%)	13.4 ± 0.2 (+54.5%)	18.5 ± 3.00 (+11.9%)

4.3.2 Decisions and Control for Main-lane and Merging-lane Vehicles

This section demonstrates the strategic control capabilities of our proposed method for main-lane and merging-lane vehicles, comparing it with the best-performing baseline to highlight its superior performance. To showcase the method's adaptability and robustness, we present data under heavy traffic density conditions. We selected MAA2C as the comparison baseline, being the second-best performer under high density. We chose two scenarios (seeds 625 and 900) to elaborate on the strategic control for main-lane and merging-lane vehicles, illustrating the advantages of our proposed method. In these scenarios, the MAA2C policy failed due to inappropriate behavior of both main-lane and merging-lane vehicles, while QMIX-QLambdaM succeeded.

Figure 4.22 and Figure 4.23 present a comparison of the two algorithms' low-level control and speed/lane change profiles for the main-lane 1 vehicle in seed 625. Figure 4.23 reveals that around time step 10, the speed in the MAA2C case drops below 10 m/s, after which the speed curve disappears, indicating a vehicle collision. Figure 4.24a depicts the scene showing the CAV colliding with the HDV, while Figure 4.24b illustrates the scene at time step 12. In the speed and acceleration profiles, the QMIX-QLambda curve exhibits frequent speed adjustments before time step 12 (frame 180) to accommodate the speed of the front HDV. It then stabilizes after time step 12, maintaining a safe gap with the preceding vehicle. In contrast, for the MAA2C case, main-lane vehicle 2 changes to the left lane, causing the HDV behind it (the predecessor of main-lane vehicle 1) to decelerate. CAV 1 attempts to slow down to accommodate the HDV's reduced speed, as evident in Figure 4.22 and Figure 4.23. However, it ultimately fails to avoid collision. Additionally, the positive and negative peaks in acceleration profiles indicate the activation of `acceleration` and `deceleration` actions respectively. As shown in Figure 4.22, prior to time step 10, QMIX-QLambdaM exhibits a total of 6 peaks, whereas MAA2C shows 7. This lower number of peaks suggests that QMIX-QLambdaM achieves smoother driving behavior.

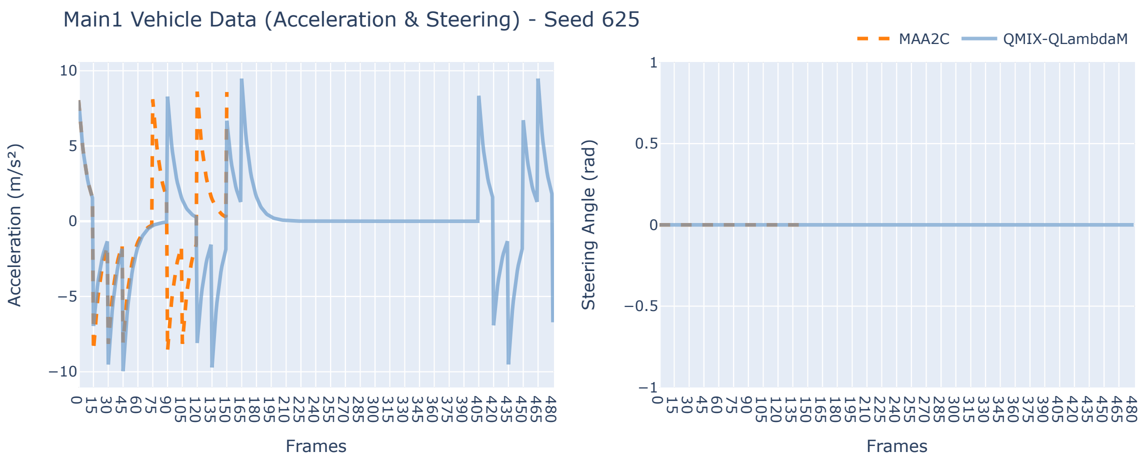


Figure 4.22: Low-level control profile of main-lane 1 vehicle of seed 625: the unit is frames as the low level control command varies every frame, 1 time step contains 15 frames

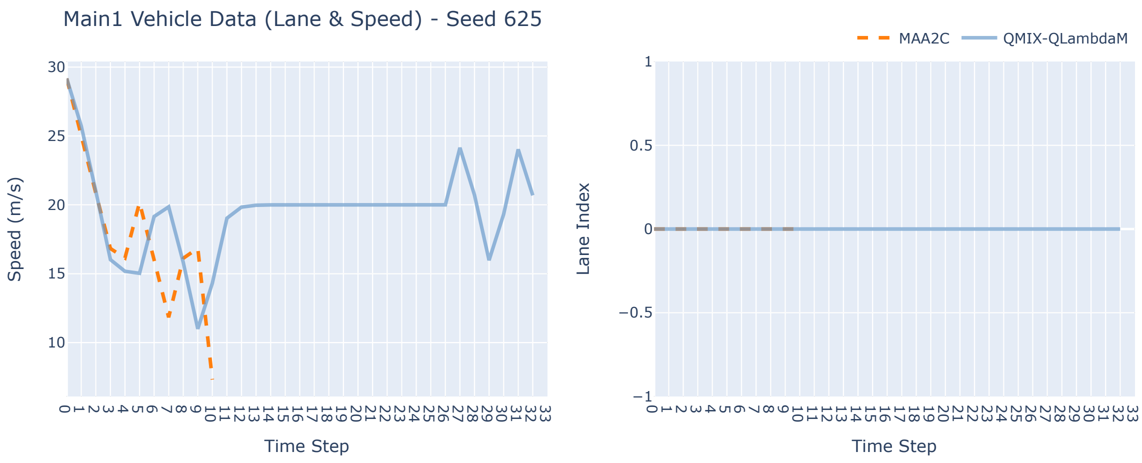
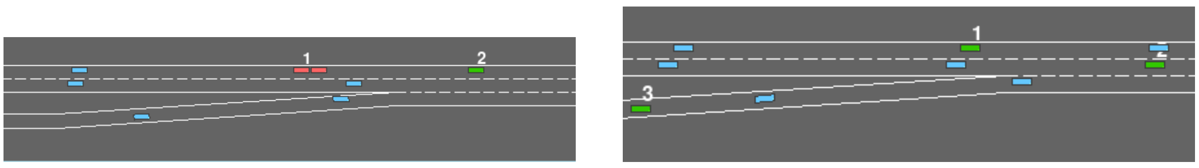


Figure 4.23: Speed and lane change profile of main-lane 1 vehicle of seed 625: the speed profile of MAA2C is lower than 10m/s at time step 10 indicating the collision



(a) The scene of collision at time step 10 for MAA2C in seed 625 (b) The scene at time step 12 for QMIX-QLambdaM in seed 625

Figure 4.24: Comparison of scenes for MAA2C and QMIX-QLambdaM in seed 625

Figure 4.26 and Figure 4.25 illustrate the low-level control commands and speed/lane changes of merging vehicles in seed 900. In Figure 4.26, lane index 3 indicates the lane prior to the merging vehicle's acceleration lane, lane index 2 indicates the merging lane, lane index 1 indicates the rightmost lane in main-lane, and lane index 0 indicates the leftmost lane in main-lane. The MAA2C curve, which terminates at approximately the 19-time steps, shows a speed below the minimum threshold with a declining trend, indicative of a collision. The lane change profile demonstrates the timing of the merging vehicle's entry into the main lane. As shown in Figure 4.26 and Figure 4.25, the MAA2C-controlled merging vehicle begins decelerating at time step 13 (frame 195), attempting to identify a suitable gap. However, at time step 19, it collides with the host HDV in the main lane, as depicted in Figure 4.27a. In contrast, the QMIX-QLambdaM-controlled merging vehicle successfully identifies a gap at time step 15 (shown in Figure 4.27b) and maintains a stable speed for the initial 4 time steps post-merge, as evidenced by the acceleration and speed profiles. Furthermore, in the Figure 4.25, before the time step 19, the total number of the peaks of the acceleration profile of QMIX-QLambdaM (8) is smaller than the value in MAA2C (17) In the steering angle profile, the proposed approach exhibits smaller magnitudes of steering angle changes during lane transitions compared to MAA2C, indicating a more stable driving behavior.

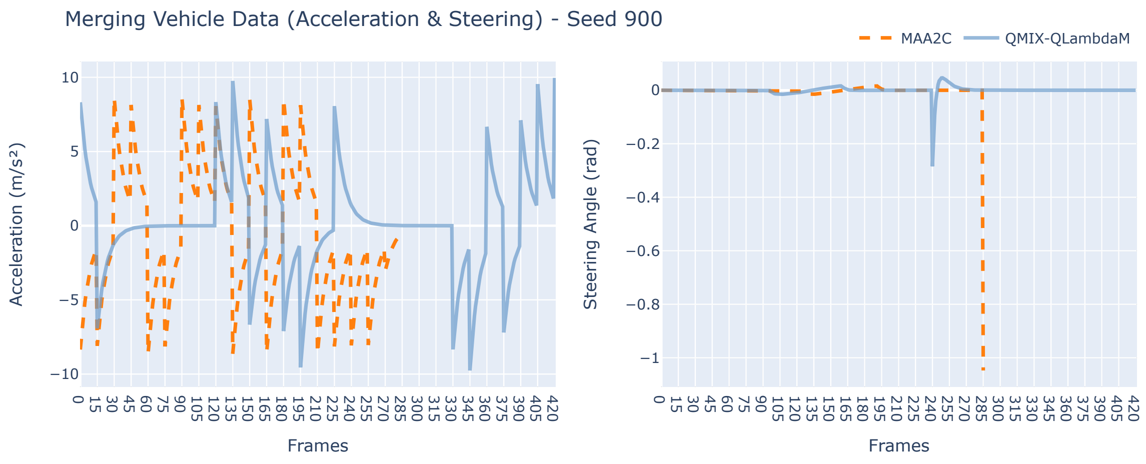


Figure 4.25: Low-level control profile of merging vehicle of seed 900

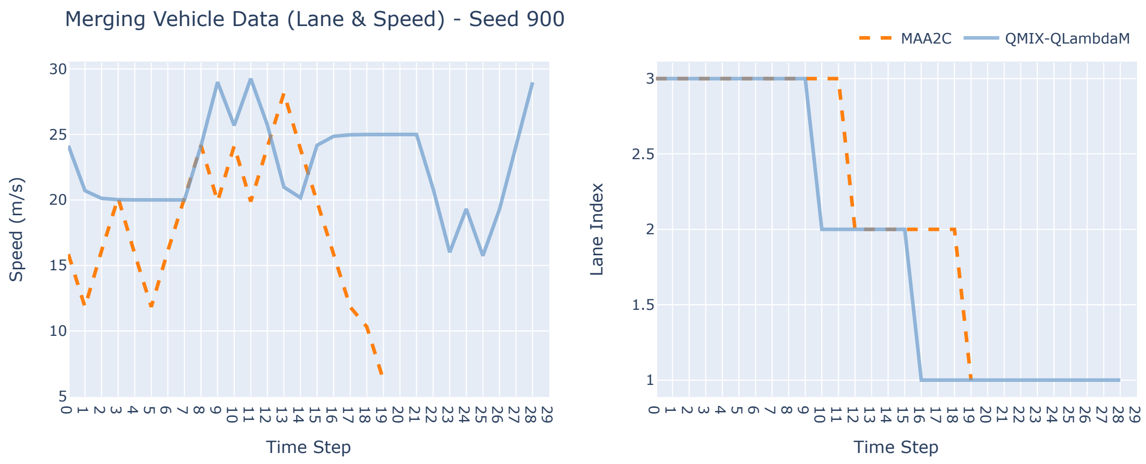
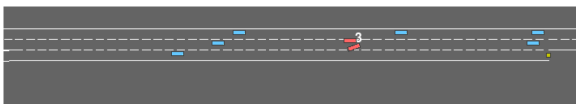


Figure 4.26: Speed and lane change profile of merging vehicle of seed 900


(a) The scene of collision at time step 19 for MAA2C in seed 900

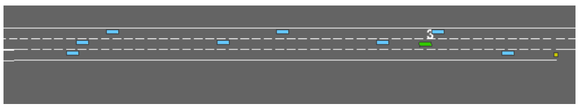

(b) The scene at time step 16 for QMIX-QLambdaM in seed 900

Figure 4.27: Comparison of scenes for MAA2C and QMIX-QLambdaM in seed 900

In conclusion, these two scenarios (seeds 625 and 900) clearly demonstrate the superior performance of QMIX-QLambdaM over MAA2C in handling complex traffic situations. In the main-lane vehicle scenario (seed 625), QMIX-QLambdaM showcased its ability to make frequent, appropriate speed adjustments to maintain safe distances, ultimately avoiding collision. Conversely, MAA2C's failure to adequately respond to changing traffic conditions led to a collision. In the merging scenario (seed 900), QMIX-QLambdaM demonstrated superior gap identification and smoother lane change execution, resulting in a successful merge without incident. MAA2C, however, failed to safely navigate the merge, resulting in a collision. These results underscore QMIX-QLambdaM's enhanced adaptability, safety, and stability in both longitudinal and lateral vehicle control.

5 Discussion and Conclusion

5.1 Discussion

In this study, I investigated the implementation of value function decomposition in cooperative on-ramp merging for CAVs. The experimental results demonstrate substantial improvements in both safety and efficiency. This section discusses the key improvements and advantages while examining the implications and limitations of the current research framework that warrant further investigation.

The learning process reveals that the proposed method demonstrates superior training efficiency and credit assignment capabilities compared to all benchmarks. However, there remains room for improvement in the learning process. One notable limitation is the lack of comprehensive hyperparameter fine-tuning, as we only performed coarse tuning through batch size adjustment. According to (J. Hu et al. 2023), the QMIX algorithm can achieve state-of-the-art performance through careful adjustment of various parameters, including the hidden dimensions of the RNN network and mixing network, replay buffer size, annealing epsilon steps, and optimizer configurations.

In the evaluation phase, the proposed method exhibits dominant performance in terms of efficiency and safety, along with superior adaptability across different traffic density levels. However, as highlighted in chapter 4, the acceleration profiles reveal a critical limitation. The maximum speed change commands generated when CAVs initiate acceleration or deceleration are unrealistic, reaching up to $10m/s^2$. This issue primarily stems from the PID low-level controller, which operates with static gains and a predetermined desired speed list featuring $5m/s$ intervals in its default configuration. Therefore, enhancing the low-level controller presents a promising direction for future improvements.

Moreover, while the action mask demonstrates significant positive impact, it still exhibits notable limitations. First of all, although the action mask shows excellent positive impact, it still has some notable limitations. It primarily focuses on vehicles either longitudinal or lateral as the ego vehicle when evaluating action safety, overlooking potential interactions with vehicles in another direction. This narrow focus can lead to incomplete safety assessments. For instance, when considering a left lane change, the mechanism doesn't account for possible movements of vehicles in the current lane that might also change lanes simultaneously. Similarly, for longitudinal commands like acceleration or deceleration, the mechanism only considers vehicles directly ahead or behind, disregarding potential lateral threats. These limitations could result in overlooking critical scenarios where vehicles from adjacent lanes pose collision risks during maneuvers.

Beyond the current findings, several future research directions merit exploration for practical applications. These include extending the approach to encompass more diverse and realistic traffic scenarios and settings, while investigating various altruistic and egoistic agents with different rewards related to social impacts. Another crucial avenue involves incorporating more sophisticated models of human driving behavior, particularly focusing on uncertain behavior models and dynamic behaviors as emphasized by (Toghi et al. 2021). Furthermore, exploring transfer learning techniques offers an promising direction for improving scalability. The current RNN-based learning architecture faces inherent limitations due to its fixed input dimension, which constrains scalability. Recent research, such as the work by (S. Hu et al. 2021), suggests that replacing RNNs with transformer architectures could effectively address this limitation, potentially enhancing the model's adaptability to varying numbers of agents and environmental complexities.

5.2 Conclusion

- **Sub Question 1:** How should the observation space, action space, and reward function be designed to balance safety and efficiency in on-ramp merging scenarios?

  Answer: The observation (state) focuses on relative positional and speed information to avoid data redundancy while maintaining efficiency. For the ego vehicle's information, the absolute position is set to zero. The action framework incorporates a high-level decision-making process with a low-level controller to facilitate speed adjustments and lane changes. The reward function is designed to be multi-objective, prioritizing safety, and is dependent on the number of collisions and the TTC, which in this project is implemented as 2D. Additionally, designating the end of the merging lane as an obstacle enables the use of TTC, but also models the merging task reward to expedite merging. Optimizing speed is also a key consideration to enhance efficiency. This design balances the need for comprehensive environmental awareness with computational efficiency, while the reward structure encourages safe and smooth merging behaviors.

- **Sub Question 2:** What impact do improvement techniques, such as Q(λ) return, have on QMIX's performance (efficiency and safety) in on-ramp merging tasks?

  Answer: Referring to (J. Hu et al. 2023), the Q(λ) return is implemented in our study. During the learning process, the Q(λ) return significantly improves the learning efficiency of QMIX. Our experiments with four granularities revealed that the optimal situation occurs when $\lambda = 0.6$. This finding suggests that a balanced approach between temporal difference learning and Monte Carlo methods yields the best results in the complex environment of on-ramp merging. The improved efficiency translates to faster convergence and more stable learning, which indirectly contributes to enhanced safety performance by allowing the algorithm to explore safe policies more effectively.

- **Sub Question 3:** How can an effective action masking mechanism be designed to improve safety and learning efficiency in QMIX for on-ramp merging?

  Answer: For safety considerations and training efficiency, an action mask mechanism has been implemented with the primary goal of excluding actions that are clearly hazardous or likely to cause collisions from the set of available actions before an agent selects its move. This approach enhances safety by reducing the likelihood of collisions. Moreover, this method effectively narrows the action space, which accelerates the learning process. The improvement in lower collision rates and higher, faster-converged learning curves relative to the original QMIX can be observed in the results section, demonstrating the efficacy of this safety-oriented strategy. This enhancement not only mitigates risk but also optimizes the system's overall efficiency by streamlining decision-making processes. The action masking mechanism proves particularly valuable in high-density traffic scenarios, where the risk of collisions is inherently higher.

- **Sub Question 4:** How does the proposed algorithm compare to other state-of-the-art MARL methods in terms of credit assignment and overall performance (efficiency and safety) in on-ramp merging scenarios?

  Answer: In the learning stage, the reward performance of QMIX-QLambdaM is superior to MAA2C and COMA, demonstrating its advantage in terms of credit assignment. This improved credit assignment allows for more effective learning in the multi-agent setting of on-ramp merging. Furthermore, QMIX-QLambdaM outperformed the state-of-the-art baselines (QMIX, COMA, MAA2C) across all measured performance indicators in the learning stage. In addition, it exhibited a faster reduction in collision rates and maintained higher average speeds throughout different experiments in various traffic densities. This comprehensive improvement is attributed to better return update means through the Q(λ) mechanism and more effective integration of safety restrictions via action masking. The algorithm's ability to maintain high performance across different traffic densities showcases its robustness and adaptability, crucial factors for real-world application in dynamic traffic environments.

The main question can be answered with the observation of all sub-questions. The main question is:
Main Question: *How can value function decomposition, specifically QMIX, be effectively implemented in cooperative on-ramp merging to enhance traffic efficiency and safety?*

This study examined the application of value function decomposition, specifically the QMIX-QLambdaM algorithm—an enhanced version of QMIX—in cooperative on-ramp merging scenarios for CAVs. The findings reveal substantial improvements in both safety and efficiency compared to existing state-of-the-art methods.

Analysis of the learning curve demonstrates QMIX-QLambdaM's superior performance, followed by the original QMIX, highlighting the potential of value function decomposition in addressing credit assignment within the formulated cooperative MARL on-ramp merging problem. Specifically, the proposed method surpasses the best baseline by a margin of 84 in terms of reward, achieves a 69.2% reduction in collision rate compared to the best baseline, and exhibits a 2.4% improvement in average speed over the best baseline. Furthermore, both improvement mechanisms contribute positively to performance enhancement. The Q(λ) return mechanism yields a 53% improvement in mean total reward compared to the original QMIX, while the action mask mechanism accelerates training through excellent initial state estimation, resulting in a 70.9% reduction in collision rate.

The evaluation results particularly highlight QMIX-QLambdaM's exceptional adaptability across varying traffic densities, demonstrating superior safety and efficiency across all three density levels. In terms of safety, the proposed method achieves a remarkable 0% collision rate across all traffic densities and maintains significant advantages over other baselines in safety rewards, demonstrating its capability to effectively prevent collisions and maintain safe distances from surrounding vehicles. Regarding efficiency, the proposed method consistently achieves a 10% advantage in average speed across all scenarios, averagely. In the context of on-ramp merging tasks, while QMIX-QLambdaM shows notable improvements over the original QMIX, it performs slightly below actor-critic methods but demonstrates superior safety in merge timing. This balance between merging efficiency and safety exemplifies the algorithm's sophisticated approach, prioritizing collision avoidance while maintaining efficient traffic flow.

In conclusion, our research demonstrates a significant evolution from QMIX's effective credit assignment to QMIX-QLambdaM's enhanced capabilities through Q(λ) returns and action masking, offering a robust solution for complex cooperative on-ramp merging scenarios. The marked improvements in safety, efficiency, and adaptability establish QMIX-QLambdaM as a promising approach for future intelligent transportation systems, particularly in scenarios requiring cooperative behavior among multiple autonomous vehicles.

6 Recommendation

The findings of this study offer valuable insights for both road authorities and vehicle manufacturers in their efforts to integrate autonomous vehicles into existing traffic systems.

6.1 Recommendations for Road Authorities

For road authorities, a key recommendation is the implementation of smart infrastructure at on-ramp merging points. Such infrastructure, capable of communicating with AVs, could significantly enhance the performance of cooperative merging algorithms like QMIX-QLambdaM by providing additional environmental data. This improvement in information flow could lead to more efficient and safer merging processes.

Road authorities should also focus on developing new traffic management policies that account for the presence of both AVs and HDVs. These policies should be designed to facilitate the kind of cooperative behavior demonstrated by our algorithm, ensuring smooth interactions between different vehicle types. Furthermore, the establishment of new safety standards and testing procedures for mixed AV-HDV traffic scenarios is crucial. The safety performance of algorithms like QMIX-QLambdaM can serve as a benchmark for these standards, helping to ensure that all vehicles on the road meet stringent safety requirements.

Another important recommendation for road authorities is the implementation of systems for collecting and sharing real-time traffic data. This data can be invaluable for further training and improving AI-based traffic management systems, leading to more adaptive and efficient traffic flow management.

6.2 Recommendations for Vehicle Manufacturers

For vehicle manufacturers, the priority should be on developing and implementing V2X communication systems in AVs. These systems are essential for enabling the kind of cooperative behavior demonstrated in this study, allowing vehicles to share information and coordinate their actions effectively. Manufacturers should also consider implementing QMIX-QLambdaM or similar cooperative algorithms as part of AVs' on-board decision-making systems, particularly for handling complex merging scenarios. This could significantly improve the ability of AVs to navigate challenging traffic situations.

The development of intuitive interfaces that can effectively communicate the intentions of AI-driven vehicles to human drivers is another crucial area for manufacturers to focus on. These interfaces could facilitate smoother interactions in mixed traffic environments, reducing confusion and potential conflicts between AVs and HDVs. Additionally, manufacturers should invest in the development of adaptive low-level control systems that can translate high-level decisions from algorithms like QMIX-QLambdaM into smooth, comfortable vehicle movements. This would enhance passenger comfort and further improve the integration of AVs into existing traffic flows.

Lastly, vehicle manufacturers should expand their simulation and real-world testing of AVs to include a wider range of mixed traffic scenarios. The findings from this study can inform test design and performance evaluation, ensuring that AVs are well-prepared for the complex and varied situations they will encounter on real roads.

By implementing these recommendations, road authorities and vehicle manufacturers can work collaboratively towards creating a safer, more efficient transportation system that smoothly integrates autonomous and human-driven vehicles, paving the way for the future of mobility.

A

Scientific Paper

Cooperative Planning and Control for Connected and Automated Vehicles' On-ramp Merging in Mixed Traffic Through Value Decomposition-based Multiagent Deep Reinforcement Learning

Yuteng Zhang

Abstract—**Connected and Automated Vehicles (CAVs) have the potential to revolutionize transportation systems, but their integration with human-driven vehicles (HDVs) in mixed traffic environments presents significant challenges, particularly in complex scenarios such as on-ramp merging. This paper addresses the challenge of on-ramp merging for CAVs in mixed traffic environments, proposing a novel approach called QMIX-QLambdaM. We formulate the problem as a Centralized Training with Decentralized Execution (CTDE) Cooperative Multi-Agent Reinforcement Learning (MARL) task, capable of handling dynamic scenarios with both CAVs and HDVs. QMIX-QLambdaM enhances the QMIX algorithm by incorporating Q(λ) returns for improved value estimation and an action masking mechanism for safer action selection. Our comprehensive experiments demonstrate that QMIX-QLambdaM consistently outperforms state-of-the-art algorithms, including QMIX, MAA2C, and COMA, across various performance metrics related to traffic efficiency and safety. The proposed method exhibits superior adaptability across different traffic densities, maintaining high performance in terms of safety, efficiency, and overall rewards. Furthermore, case studies illustrate QMIX-QLambdaM's ability to generate effective strategic control for both main-lane and merging-lane vehicles, showcasing smoother driving behavior and better collision avoidance compared to baseline methods. The learning curve comparison also reveals QMIX-QLambdaM's advantage in credit assignment compared to other CTDE baselines for the formulated problem. The code are available at https://github.com/ayton-zhang/MARL_qmix_merging.**

Index Terms—**Multi-agent Reinforcement Learning, Deep Reinforcement Learning, On-ramp Merging, Value Function Decomposition.**

I. Introduction

AUTONOMOUS Vehicles (AVs) have made significant strides in commercial applications, with notable examples such as Baidu Apollo and Waymo demonstrating upward signs of progress in the real world. Furthermore, advancements in cutting-edge communication technologies, including vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-cloud (V2C) systems (Dey et al., 2016), have paved the way for Connected Autonomous Vehicles (CAVs) to become a viable reality. These CAVs can access more comprehensive surrounding information, such as the position and speed of nearby road participants, thereby enhancing collective decision-making capabilities and ultimately improving traffic efficiency and safety. However, while the potential of fully autonomous transportation systems is compelling, current projections indicate that only 20-40% of vehicles are expected to be automated by 2030, with full penetration likely achievable in a few decades (Zong, 2019). Consequently, the present reality poses a more nuanced challenge: the mixed traffic environment. In this context, AVs must not only react to road objects but also adapt to and anticipate the behaviors of human-driven vehicles (HDVs).

Among the various applications of CAVs, the on-ramp merging scenario has garnered increasing attention due to its critical role in traffic management. The complexity of on-ramp merging lies in its demand for precise coordination between multiple vehicles. Vehicles in the main lane near the merging point must dynamically adjust their speeds to create sufficient space for incoming vehicles. Simultaneously, vehicles on the on-ramp face the challenge of calibrating their speed and executing timely lane changes to seamlessly integrate into the main traffic flow, all while avoiding potential deadlocks (Irshayyid et al., 2024). It would also be more challenging when the AVs had to deal with the erratic HDVs in the mixed traffic environment.

Traditional methods such as the rule-based methods and optimization-control methods have been fully investigated. For the rule-based methods, they rely on predefined, hard-coded rules (Schmidt and Posch, 1983), (Uno et al., 1999), (Marinescu et al., 2012). (Min et al., 2021) noted that while these methods may work in controlled environments, they struggle with the variability of real traffic conditions, such as unexpected congestion or accidents, which require more dynamic and responsive control strategies. In contrast, optimization-based control strategies, such as Model Predictive Control (MPC), employ detailed dynamic models to represent vehicle interactions (Athans, 1969), (Cao et al., 2015),(Sun et al., 2020). (Irshayyid et al., 2024) suggest that while these methods show promise, they have significant drawbacks. They rely heavily on precise modeling, even for unpredictable HDVs, and often require substantial computational resources due to solving complex optimization problems at each time step.

Another line of work is to use data-driven methods and explore their effects in on-ramp merging scenarios, especially Deep Reinforcement Learning (DRL). DRL's strength lies in its ability to learn optimal decision-making policies through continuous interaction with the environment without the precise model of the system, which allows the agent to adapt to changing conditions and unpredictable behaviors, addressing many of the challenges faced by traditional methods. Initial research predominantly focused on single-agent reinforcement learning, where one AV interacts with multiple HDVs. (Triest et al., 2020) utilized over 400 real highway merging scenarios from the NGSIM dataset to train and test a single ego vehicle. In their setup, surrounding vehicles followed pre-recorded trajectories from the NGSIM data, not responding to the ego vehicle's actions. Similarly, (Wang and Chan, 2017) employed a single-agent approach, combining LSTM and DQN to learn optimal policies while addressing challenges like balancing exploration/exploitation and avoiding local optima. (Bouton et al., 2019) also adopted a single-agent perspective, focusing on the problem of uncertainty in predicting whether other vehicles will cooperatively create gaps or not. They simulated an ego vehicle approaching a merge point and interacting with up to 16 randomly behaving surrounding vehicles, some set as cooperative and others as non-cooperative.

Despite the advancements in single-agent reinforcement learning, these approaches have a fundamental limitation when applied to on-ramp merging scenarios. By controlling only one agent and treating other vehicles as part of the static environment, they fail to capture the inherent need for coordination between multiple vehicles, particularly between those on the main highway and those merging from the ramp. By addressing this issue, the paradigm is extended to MARL where multiple agents interact with a shared environment. Normally, the CAVs on-ramp merging problem always modeled as cooperative MARL problem (Toghi et al., 2021), (Valiente et al., 2022), (Sumanth Nakka et al., 2022), (Chen et al., 2023), (Hu et al., 2019), where all agents share the unified reward.

However, in the cooperative MARL, credit assignment is crucial as algorithm updates rely on a centralized critic and consider unified rewards. When multiple agents collaborate to optimize a shared reward, accurately determining each agent's individual contribution becomes problematic. Sometimes, bad credit assignment would cause the lazy agents problem which would deteriorate the convergence performance (Du et al., 2023).

To address the challenge of credit assignment in MARL, existing methods in the realm of CAVs on-ramp merging can be broadly classified into two principal approaches: Decentralized Training with Decentralized Execution (DTDE) and Centralized Training with Decentralized Execution (CTDE). Table I summarizes key papers on Cooperative MARL for CAVs in on-ramp merging, detailing algorithms, credit assignment approaches, action and observation spaces, and reward structures. The DTDE approach aims to accurately capture the individual contributions of different agents by either utilizing local rewards directly or decomposing global rewards into local components, rather than relying on a unified reward system. (Chen et al., 2023) developed an decentralized MAA2C framework that features a priority-based safety supervisor. Their approach integrates action masking and parameter sharing to foster inter-agent cooperation. In addition, they used the local reward to capture the credit assignment. (Toghi et al., 2021) focused on developing cooperative and altruistic behaviors in CAVs using decentralized MAA2C. Their research simulates CAVs learning to coordinate with each other and yield to human-driven merging vehicles, with a particular emphasis on fostering altruistic behaviors in mixed traffic environments. They proposed a novel reward structure that encourages CAVs to consider the interests of other vehicles, promoting a more harmonious traffic flow. Their papers use decentralized reward structures where each agent optimizes its own reward function which is related to credit assignment. (Valiente et al., 2022) proposes a decentralized MARL framework for training cooperative CAVs in mixed-autonomy traffic. The approach uses a 3D Convolutional Neural Network with a safety prioritizer and a novel decentralized reward function that accounts for social utility. The framework addresses credit assignment through mechanisms like decentralized rewards, Social Value Orientation, semi-sequential training, and experience replay. Although the DTDE utilizes decentralized reward or local reward capturing the credit assignment, to mitigate the issue of non-stationarity (Hernandez-Leal et al., 2019), DTDE methods typically require training different agents either in isolation or in predetermined sequences (Irshayyid et al., 2024) leading to inefficient training process.

In contrast to DTDE, the CTDE paradigm has emerged as a promising approach to address the non-stationarity challenge inherent in multi-agent systems. This framework aims to leverage the benefits of centralized information during the training phase while maintaining the decentralized nature of execution. In the field of on-ramp merging for CAVs, the CTDE framework has gained significant traction. (Sumanth Nakka et al., 2022) introduced a comprehensive framework using MADDPG approach that employs multi-objective optimization, simultaneously considering energy efficiency and travel time for which there is not a mechanism geared for credit assignment. (Hu et al., 2019) introduced the IDAS model, which employs a modified COMA method combined with curriculum learning. They used a counterfactual baseline in their centralized critic to address the credit assignment issue. The counterfactual compares the joint Q-value to the Q-value obtained if the agent did not take the action, which helps calculate the agent's contribution to the overall performance.

Reference	Algorithm	Paradigm	Credit Assignment	Action	Observation	Reward
(Chen et al., 2023)	Decentralized MAA2C	DTDE	local reward	(decelerate, keep lane, accelerate, turn left, turn right)	(speed, position)	collision & speed & headway & merging waiting
(Toghi et al., 2021)	Decentralized MAA2C	DTDE	decentralized reward	(decelerate, keep lane, accelerate, turn left, turn right)	multi-channel speed position information	speed, lane change, altruistic
(Valiente et al., 2022)	DDQN	DTDE	decentralized reward	(decelerate, keep lane, accelerate, turn left, turn right)	multi-channel dynamics information	egoistic and altruistic reward
(Sumanth Nakka et al., 2022)	MADDPG	CTDE	-	acceleration	(position, speed)	rear-end safety & lateral safety & vehicle energy consumption
(Hu et al., 2019)	COMA	CTDE	counterfactual baseline	(high decelerate, decelerate, maintain-speed, accelerate, high accelerate)	(road priority, driver type, current lane occupancy, other lane occupancy)	finish & collide & flow& impede

TABLE I: Cooperative MARL CAVs On-ramp Merging

Although various methods have been implemented, this review still reveals a significant gap: the absence of value decomposition methods in cooperative on-ramp merging research. Value function decomposition method is specifically designed to tackle the problem of credit assignment in cooperative MARL. In value function decomposition method, to ensure effective coordination, a centralized action-value function, Q_{tot}, is developed, encapsulating the collective benefits achievable by the entire system. However, to direct individual agents with a decentralized policy, it is crucial to accurately attribute portions of Q_{tot} to each agent's contribution. The core principle of value function decomposition lies in breaking down a joint value function into separate value functions for each agent, thereby isolating and recognizing the unique contributions of each agent towards cooperative objectives. The pioneer work is VDN which employs a linear combination of individual Q value functions, which is suitable for simpler scenarios (Sunehag et al., 2018). To address more complex problems, QMIX introduces a monotonicity constraint that enhances the representational capacity of the model (Rashid et al., 2020b). Based on QMIX, different kinds of constraints are investigated (Son et al., 2019), (Rashid et al., 2020a). Notably, the study by (Hu et al., 2023) demonstrates that through specific code-level fine-tuning techniques, QMIX can be enhanced to achieve state-of-the-art performance in the StarCraft Multi-agent Challenge (SMAC) task (Samvelyan et al., 2019), which shows the advantage over other kinds of constraints in the mixing network combination.

In mixed-traffic on-ramp merging scenarios, ensuring collision-free operations is paramount. Researchers have developed various safety-enhancing approaches within MARL frameworks to address this critical requirement. (Chen et al., 2023) introduced a rule-based safety supervisor that projects vehicle trajectories for future time steps based on the current policy, ensuring that only safe actions are executed. To improve learning efficiency and safety simultaneously, (Hu et al., 2019) implemented an action masking mechanism. This technique prevents autonomous vehicles from exploring implausible or unsafe states, effectively reducing the action search space while enhancing overall safety. Drawing inspiration from these approaches, our algorithm incorporates a similar action masking mechanism to enhance safety while maintaining efficient learning.

The key contributions of our work are:

- In our study, we approach the on-ramp merging challenge in mixed traffic conditions as CTDE MARL. This mixed traffic scenario encompasses both AVs and HDVs operating concurrently on both the on-ramp and mainline lanes. The modeling framework we've developed is specifically tailored to handle dynamic scenarios, accommodating fluctuations in the quantity and spatial distribution of HDVs, as well as variations in the positioning of CAVs.
- We developed the QMIX-QLambdaM (an improved version of QMIX) for cooperative mixed traffic on-ramp merging problem with a multi-objective reward function. The QMIX-QLambdaM improved the performance of QMIX by incorporating Q(λ) return for improved value estimation and an action masking mechanism to expedite learning and enhance safety.
- Our extensive experimental evaluations demonstrate that the novel approach we've developed consistently achieves superior performance compared to existing cutting-edge algorithms. This superiority is evident in both safety metrics and operational efficiency during the training process. Furthermore, our method exhibits enhanced adaptability across varying traffic density scenarios.
- In the context of our formulated on-ramp merging problem, our analysis reveals that QMIX-QLambdaM demonstrates superior credit assignment capabilities when compared to other state-of-the-art CTDE baseline algorithms.

II. Preliminaries

A. Decentralized POMDP

In addressing the challenge of multi-vehicle driving, we could recognize that each agent operates with limited individual perception and shared unified reward. To model the *cooperative multi-agent task*, the problem is specifically modeled as a decentralized partially observable Markov decision process (Dec-POMDP) model. This model is succinctly represented by the tuple:

$$G = \langle S, A, P, r, Z, O, n, \gamma \rangle$$

Here, $s \in S$ denotes the actual state of the environment, with the Dec-POMDP framework accounting for situations where each agent receives its own incomplete observation $z \in \mathcal{Z}$, derived from the observation function $O(s, a) : S \times A \rightarrow [0, 1]$. Within each time step, an agent i from the set $N \equiv 1, \ldots, n$ selects an action $a_i \in A_i$ guided by its policy $\pi_i(a_i \mid \tau_i) : \mathcal{T} \times \mathcal{A} \rightarrow [0, 1]$, where $\tau_i \in \mathcal{T} := (\mathcal{Z} \times \mathcal{A})^*$ represents the agent's history of actions and observations. These individual actions combine into a collective action $a \in \mathcal{A} \equiv A'$, leading to the next state according to the state transition function $P(s' \mid s, a) : S \times \mathcal{A} \times S \rightarrow [0, 1]$. A unified reward function $r(s, a) : S \times \mathcal{A} \rightarrow \mathbb{R}$ benefits all agents, with the action-value function defined as $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}[R_t \mid s_t, a_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ represents the reward accumulated over time, discounted at rate γ. While the training process is conducted in a centralized manner, allowing the learning algorithm to utilize both the collective action-observation histories τ and the overall state s, the execution phase is decentralized.

B. Deep Q-learning

Deep Q-Learning extends traditional Q-Learning by incorporating deep neural networks to approximate the action-value function $Q(\cdot)$. This approach, pioneered by (Mnih et al., 2015), addresses the limitations of tabular methods in handling high-dimensional state spaces.

The algorithm employs a neural network, parameterized by θ, to estimate $Q(s, a)$ for all state-action pairs. To enhance training stability, it utilizes an experience replay buffer $\mathcal{D}$, which stores transitions (s_t, a_t, r_t, s_{t+1}) encountered during interaction with the environment.

During training, mini-batches of size B are randomly sampled from $\mathcal{D}$ to update the network parameters. The loss function $\mathcal{L}(\theta)$ is defined as the mean squared error between the current Q-value estimates and the target Q-values:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[(y_i - Q(s_i, a_i; \theta))^2 \right] \tag{1}$$

where the target value y_i is computed as:

$$y_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta^-) \tag{2}$$

Here, θ^- represents the parameters of a target network, which is periodically updated to match the main network. This technique, known as fixed Q-targets, further stabilizes the learning process by reducing the moving target problem.

C. QMIX

The decomposition method of the value function is designed to address the challenge of credit assignment by factorizing the joint Q-value function into a combination of individual Q-value functions. This factorization distinguishes the unique contributions of each agent, which is realized by Individual-Global-Max (IGM) principle (Son et al., 2019). This principle ensures that the optimal joint action, obtained through the global maximization of the team's Q function value, aligns with the aggregation of Q function values that have been maximized individually, which is mathematically expressed in Equation 3.

$$\arg\max_{a} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \arg\max_{a_N} Q_N(\tau_N, a_N) \end{pmatrix} \tag{3}$$

In QMIX, to ensure the condition of Equation 3 hold, additionally, a constraint Equation 4 is imposed on both the collective and individual value functions to guarantee their monotonicity, ensuring that the integrated value function consistently reflects the aggregated impact of each agent's actions.

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A \tag{4}$$

The architecture of the QMIX algorithm is shown in Figure 3, which features two critical components: the agent network and the mixing network. The agent network receives as input the individual observations and previous actions of each agent. Its configuration employs recurrent neural networks (RNNs) to enable agents to leverage their complete history of actions and observations. Essentially, the agent network generates an estimated action value Q_i for agent i, which is subsequently relayed to the mixing network. The mixing network incorporates the overall state of the environment along with the estimated action values as input. The parameters of the mixing network are generated through a set of dedicated hypernetworks. Each hypernetwork receives the state s as input and produces weights for one layer of the mixing network. These hypernetworks have a relatively simple structure, consisting of a single linear layer followed by an absolute value activation function. The purpose of using an absolute value activation is to ensure that the generated mixing network weights are all non-negative. The output of each hypernetwork is a vector, which is subsequently reshaped into a matrix of appropriate dimensions. The process for generating bias terms is similar to that of weights, but without the non-negativity constraint. Notably, the final bias term is produced by a slightly more complex hypernetwork, which incorporates two layers with a ReLU non-linearity between them. The primary goal is to optimize the network by minimizing the specified loss function.

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[(y_i - Q_{tot}(\tau, u, s; \theta))^2 \right] \tag{5}$$

The loss function, as referenced in Equation 5, aligns with the traditional DQN algorithm's framework, with the distinction that the Q value in this context is a cumulative Q_{tot} value. Throughout the training phase, every agent implements an ϵ-greedy strategy, where ϵ regulates the equilibrium between exploration and exploitation for each agent, gradually decreasing as training progresses.

III. Methodology

A. RL Formulation

1) State and Observation Space:: The state space in reinforcement learning for autonomous driving should balance essential information with computational efficiency. It typically includes the ego vehicle's kinematics and data on surrounding traffic. Our approach focuses on immediately adjacent vehicles to maintain training efficiency while capturing crucial environmental dynamics (Irshayyid et al., 2024). Each agent i receives observations as a matrix o_i with dimensions $N_i \times W$. N_i represents observable vehicles within 150 meters (Yu et al., 2020), usually up to six: front, rear, left front, left rear, right front, and right rear. W encompasses various vehicle attributes:

- $ispresent$: This binary variable denotes whether a vehicle occupies this feature position within the observable surroundings.
- x : Represents the relative horizontal displacement of surrounding vehicles with respect to the ego vehicle. For the ego vehicle itself, this value is set to zero to optimize training efficiency. While this information is primarily relevant for evaluating lane change rewards (discussed later), the pertinent data is captured in the last action feature $\bar{u}$.
- y : Indicates the relative vertical displacement of surrounding vehicles in relation to the ego vehicle. For the ego vehicle, this is set to 0, similar to x.
- vx : Represents the relative horizontal speed of surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the absolute horizontal velocity.
- vy : Denotes the relative vertical speed of surrounding vehicles in relation to the ego vehicle. For the ego vehicle, this is the absolute vertical velocity.
- cos_h : Represents the cosine of the relative heading angle for surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the cosine of its absolute heading.
- sin_h : Indicates the sine of the relative heading angle for surrounding vehicles with respect to the ego vehicle. For the ego vehicle, this is the sine of its absolute heading.
- $\bar{u}$: Denotes the action executed by the agent in the previous time step.

The comprehensive state space is formed by amalgamating the observations of all individual agents, denoted as $s = [o_i]_{i=1}^{N}$, where N represents the total number of controlled CAVs.

2) Action Space:: The range of actions available to each agent comprises several high-level strategic maneuvers such as $lane\ change\ left$, $lane\ change\ right$, $maintain\ lane$, $accelerate$, and $decelerate$ for training efficiency without sacrificing the essence of driving behavior. These strategies are implemented through a lower-level control system, which adjusts the steering and throttle based on the selected action. The collective action space for all the CAVs in the system is a composite of the individual actions, represented as $A = A_1 \times A_2 \times \cdots \times A_N$.

3) Reward Function:: The reward function r_i plays a pivotal role in shaping agent behavior towards desired outcomes in our autonomous driving system. Our design focuses primarily on two critical objectives: safety through collision avoidance and efficiency via speed optimization. To support these main goals, we incorporate several secondary factors into our reward structure. The reward r_i for each agent i is formulated as follows:

$$r_i = w_c r_c + w_s r_s + w_{ttc} r_{ttc} + w_l r_l + w_m r_m \tag{6}$$

where w_c, w_s, w_{ttc}, w_l, and w_m are positive weights for the penalty of collision, speed reward, time-to-collision (TTC) evaluation, penalty for change of lanes, and the merging task reward. The five performance indicators are defined as follows in more detail:

- the r_c of the agent i is set to be -1 if the vehicle has a collision, otherwise it is 0.
- the r_s is defined as

$$r_s = \min\left\{\frac{v_t - v_{\min}}{v_{\max} - v_{\min}}, 1\right\} \tag{7}$$

  The settings for $v_{\min} = 10m/s$ and $v_{\max} = 30m/s$ are based on the configurations described in (Chen et al., 2023). These parameters were chosen by considering two key sources: recommendations from the (US Department of Transportation, 2018) and empirical data from the Next Generation Simulation (NGSIM) dataset (Thiemann et al., 2008). The upper limit aligns with the transportation authority's suggested speed range ($20 - 30m/s$). At the same time, the lower bound takes into account the minimum speeds observed in real-world traffic scenarios, as captured in the NGSIM data (minimum speed at $6 - 8m/s$).
- As the vehicle can observe up to 6 surrounding vehicles, it is crucial to determine the appropriate number of vehicles to consider in the TTC evaluation. The reward associated with TTC, r_{ttc}, is defined in Equation 8. Furthermore, a variable N represents the set of evaluation vehicles, ordered from the smallest to the largest TTC. The possible values for N range from 1 to 6. The calculation of the TTC value follows the methodology described in (Jiao, 2023), which accounts for both longitudinal and lateral TTCs. The evaluation function for the TTC value is structured as follows: the safe TTC threshold, TTC_{safe}, is set at $2s$ (Kruber et al., 2019). Since TTC values of ∞ and -1 are not applicable within the function, these are substituted with $12s$ and $1e-9$, respectively (Kruber et al., 2019). The corresponding trend for this function is depicted in Figure 1, demonstrating that when the TTC is below the safe threshold, the reward decreases significantly.

$$r_{ttc} = \frac{1}{N}\sum_{n=1}^{N} \log\left(\frac{TTC_n}{TTC_{safe}}\right), \quad TTC_n \in [1e{-}9, 12], \quad n \in N \tag{8}$$

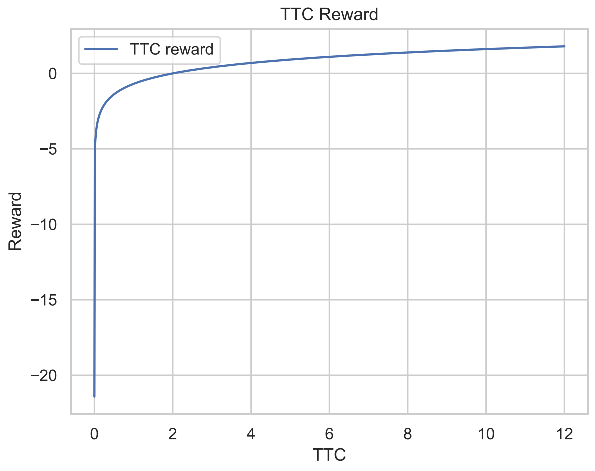


Fig. 1: TTC Reward

- For agent i, the lane change reward, r_l, is set to -1 if the vehicle is not merging vehicle, otherwise, the reward is zero.
- The merging task reward for agent i is formulated as a function of the distance traveled on the ramp, denoted as x. This cost is designed to encourage timely merging and is defined as:

$$r_m = \frac{1}{40000}x^2 - \frac{1}{100}x, \quad x \in [0, l_{ramp}] \tag{9}$$

where l_{ramp} represents the total length of the merging ramp. Figure 2 illustrates the relationship between the merging cost and the distance traveled. This quadratic function ensures that the cost increases more rapidly as the vehicle approaches the end of the merging lane, thereby discouraging prolonged waiting in the merging area.

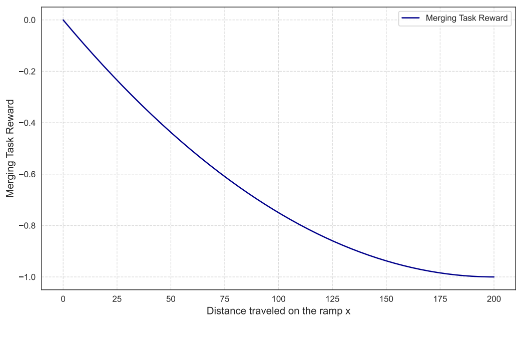


Fig. 2: Merging Task Reward

B. QMIX-QLambdaM: Improved QMIX with Q Lambda Returns and Action Masking

In this section, we present QMIX-QLambdaM, our proposed enhancement to the QMIX algorithm for cooperative on-ramp merging scenarios. QMIX-QLambdaM incorporates two key improvements: the $Q(\lambda)$ return for more efficient value estimation, and an action masking mechanism for safer action selection which is detailed in Figure 3.

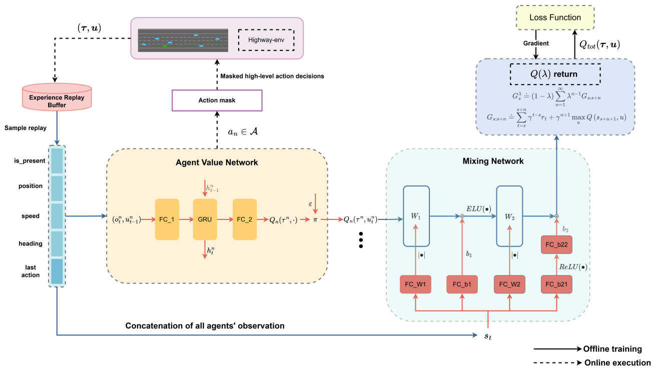


Fig. 3: The structure of QMIX-QLambdaM

1) Q(λ) Return:: Our approach incorporates eligibility traces based on the Peng's Q(λ) method (Peng and Williams, 1994). This technique has been empirically shown to enhance the stability and convergence speed of QMIX, while mitigating the significant bias against bootstrap returns that can arise from insufficiently trained Q-networks (Hu et al., 2023). The Q(λ) formula for the mixing network is defined as follows:

$$G_s^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{s:s+n} \tag{10}$$

where $G_{s:s+n}$ is computed as:

$$G_{s:s+n} = \sum_{t=s}^{s+n}\gamma^{t-s}r_t + \gamma^{n+1}\max_a Q_{tot}(\tau, a, s_t; \theta_{tot}) \tag{11}$$

In these equations, G_s^λ replaces the target value y_{target} in the original loss function. The parameter λ serves as the discount factor for the traces, with the property that $\prod_{s=1}^{t}\lambda = 1$ when $t = 0$.

2) Action Masking Mechanism:: To enhance safety and accelerate training, QMIX-QLambdaM employs an action masking mechanism that constrains the action selection process to a subset of reasonable and safe actions. This approach not only prevents actions that could lead to accidents or traffic rule violations but also expedites the learning process by effectively reducing the action space. The mechanism, integral to QMIX-QLambdaM's design, is implemented during the action selection phase.

The action masking mechanism in QMIX-QLambdaM comprises two key filtering algorithms: `LongitudinalMask` and `LateralMask`, detailed in Algorithms 1 and 2, respectively. These algorithms ensure the safety of both speed change and lane change commands. It's worth noting that actions violating speed limits or road topology constraints are already masked by built-in functions in the simulator. In the pseudocode, italics represent variables, while bold text indicates algorithms and equations. At its core, this mechanism operates by simulating each potential action for one time step before the agent's selection, allowing for a proactive safety assessment.

In the `LongitudinalMask`, the inputs are the front vehicle and the rear vehicle. Initially, both are processed by a function called `UnsafeVehicle`, which evaluates the longitudinal distance between two vehicles to determine if further assessment is necessary. If a surrounding vehicle is deemed safe, it is set to None. Next, the activation of `AccelerateMask` and `DecelerateMask` depends on the presence of v_f and v_r, which represent the front and rear vehicles, respectively. The main function of `AccelerateMask` is to simulate, for the next time step, the ego vehicle executing $acceleration$ or $idle$, while the corresponding surrounding vehicle, depending on its type, executes its own action. If v_{surr} is a controlled vehicle, it will execute the most dangerous action relative to the ego vehicle (in this case, v_f executing $deceleration$). However, if it is a human-driven vehicle, it will follow the IDM policy. The difference between vehicle types is that for controlled vehicles, the masking mechanism will override their policy, whereas human-driven vehicles will always adhere to their policy. The simulation lasts for one step, as the action frequency is one action per time step. During the simulation, if the TTC value (note that this is one-dimensional, as the $2DTTC$ is computationally intensive) between the ego vehicle and the surrounding vehicle is shorter than the TTC_{safe}, the executed action for the ego vehicle will be masked out in the available action space. It's important to note that masking the $idle$ action will only occur when actions other than $acceleration$ and $idle$ are available, to prevent an empty action space. The logic is similar for the `DecelerateMask`. When both v_f and v_r are unsafe regarding the ego vehicle, their TTC values will be compared to determine which poses the greater danger to determine which mask will be activated.

In the `LateralMask`, decision-making focuses on whether to filter out actions for turning left or right. Algorithm 2 details this process. The vehicles v_{lf}, v_{lr}, v_{rf}, and v_{rr} are processed through `UnsafeVehicle` to assess their safety. Subsequently, the activation of `LeftMask` and `RightMask` depends on the presence of corresponding surrounding vehicles. This logic is more intricate compared to the `LongitudinalMask`, as identifying the most hazardous scenario for the ego vehicle depends on the relative position of the surrounding vehicles. In the `LeftMask`, for instance, one case considers a left vehicle (a CAV) not in the immediately adjacent lane, assumed to turn right. This represents the most threatening situation when the ego vehicle intends to turn left. Conversely, if the left vehicle is in the adjacent lane and is a CAV, depending on its relative longitudinal position to the ego vehicle, the most dangerous action v_{lf} and v_{lr} are $acceleration$ and $deceleration$, respectively. Unlike the longitudinal scenario, if a collision is detected during these maneuvers, the option to change lanes left is filtered out of the available actions. The logic is also similar to `RightMask`. Consequently, up to two surrounding vehicles are considered in both `LeftMask` and `RightMask`.

Algorithm 1 LongitudinalMask

```
1: function LONGITUDINALMASK($ego$, $v_f$, $v_r$, $available\_actions$, $t$)
2:     $v_f \leftarrow$ UNSAFEVEHICLE($ego$, $v_f$)
3:     $v_r \leftarrow$ UNSAFEVEHICLE($ego$, $v_r$)
4:     if $v_f \neq$ None and $v_r =$ None then
5:         $available\_actions \leftarrow$ ACCELERATEMASK($ego$, $v_f$, $available\_actions$, $t$)
6:     end if
7:     if $v_r \neq$ None and $v_f =$ None then
8:         $available\_actions \leftarrow$ DECELERATEMASK($ego$, $v_r$, $available\_actions$, $t$)
9:     end if
10:    if $v_f \neq$ None and $v_r \neq$ None then
11:        $TTC_f \leftarrow$ 1DTTC($ego$, $v_f$)
12:        $TTC_r \leftarrow$ 1DTTC($ego$, $v_r$)
13:        if $TTC_f \leq TTC_r$ then
14:            $available\_actions \leftarrow$ ACCELERATEMASK($ego$, $v_f$, $available\_actions$, $t$)
15:        else
16:            $available\_actions \leftarrow$ DECELERATEMASK($ego$, $v_r$, $available\_actions$, $t$)
17:        end if
18:    end if
19:    return $available\_actions$
20: end function
```

Algorithm 2 LateralMask

```
1: function LATERALMASK($ego$, $v_{rf}$, $v_{rr}$, $v_{lf}$, $v_{lr}$, $available\_actions$, $t$)
2:     $v_{lf} \leftarrow$ UNSAFEVEHICLE($ego$, $v_{lf}$)
3:     $v_{lr} \leftarrow$ UNSAFEVEHICLE($ego$, $v_{lr}$)
4:     $v_{rf} \leftarrow$ UNSAFEVEHICLE($ego$, $v_{rf}$)
5:     $v_{rr} \leftarrow$ UNSAFEVEHICLE($ego$, $v_{rr}$)
6:     if $v_{lf} \neq$ None or $v_{lr} \neq$ None then
7:         $available\_actions \leftarrow$ LEFTMASK($ego$, $v_{lf}$, $v_{lr}$, $available\_actions$, $t$)
8:     end if
9:     if $v_{rf} \neq$ None or $v_{rr} \neq$ None then
10:        $available\_actions \leftarrow$ RIGHTMASK($ego$, $v_{rf}$, $v_{rr}$, $available\_actions$, $t$)
11:    end if
12:    return $available\_actions$
13: end function
```

IV. Experiment and Result Analysis

A. Experimental Setting

Figure 4 illustrates the road structure and vehicle spawn points for experiments. Table II provides comprehensive details about the simulation and experimental parameters. The simulation is conducted using the `highway-env` simulator (Leurent, 2018). To enhance realism, position noise is added at the spawn points, introducing variability in vehicle starting positions. Vehicles on the main lane begin moving at speeds between 25-27 m/s, while those on the merging lane start between 20-22 m/s, in accordance with (Dutch Design Guidelines, 2024). The simulation enforces an action frequency of one second, meaning that the agent takes one action per second of simulation time. Although vehicles may exit the depicted road structure, their dynamics continue to be computed within the simulation framework. The distribution of HDVs and CAVs is also outlined in Table II. The chosen distribution of CAVs, with one CAV per lane, allows for the capture of different behaviors based on vehicle position while keeping the learning process as simple as possible to reduce computational burden. Two HDVs are placed in each main lane to enable CAVs to learn adaptive policies, as they may spawn between the two HDVs. The merging lane contains one HDV, as the number of vehicles evaluated in the TTC reward is variable and up to six. Consequently, the merging lane contains at least two vehicles.

The `highway-env` employs a simplified vehicle dynamics model based on bicycle kinematics (Polack et al., 2017) to represent vehicle motion. A proportional-integral-derivative (PID) controller converts high-level commands into specific steering and acceleration inputs. For HDVs, the environment incorporates behavioral models for both lateral movement and longitudinal control, drawing inspiration from (Treiber et al., 2000) and (Kesting et al., 2007).

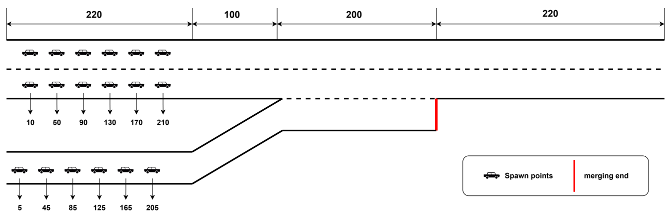


Fig. 4: Simulation Scenario

Parameters	Value
Simulator	Highway-env
Total Length	740
Entrance of Merge	320
Merge	200
Exit of Merge	220
Position Noise	[-1.5, 1.5]
Initial Speed of Main-lane	25 - 27 m/s
Initial Speed of Merging-lane	20 - 22 m/s
Action Frequency	1 seconds
Number of CAV in First Main-lane	1
Number of CAV in Second Main-lane	1
Number of CAV in Merging-lane	1
Number of HDV in First Main-lane	2
Number of HDV in Second Main-lane	2
Number of HDV in Merging-lane	1

TABLE II: Parameters of Simulation

B. Comparison of the Learning Curve with State-of-the-art Benchmarks

This section compares our proposed algorithm, QMIX-QLambdaM, with three state-of-the-art (SOTA) CTDE baselines: QMIX, MAA2C, and COMA. MAA2C extends the A2C (Advantage Actor-Critic) algorithm to multi-agent settings by introducing a centralized critic to evaluate joint rewards, enabling simultaneous training of agents using decentralized actors. COMA builds upon MAA2C by addressing credit assignment through a counterfactual baseline that marginalizes out individual agent actions while keeping others fixed. The learning parameters for QMIX and QMIX-QL ambdaM in Table III are adapted from (Rashid et al., 2018) with some coarse tuning, while MAA2C and COMA parameters in Table IV are based on (Foerster et al., 2017). For QMIX-QLambdaM, we set λ to 0.6 which has the best learning performance.

TABLE III: The learning parameters of QMIX and QMIX-QLambdaM which are borrowed from (Rashid et al., 2018), λ is only used for QMIX-QLambdaM

Parameters	QMIX-QLambdaM & QMIX
FC_1	[15(7+5+3), 64]
GRU	[64, 64]
FC_2	[64, 5]
FC_W1	[45(5*15), 96(3*32)]
FC_b1	[45, 32]
FC_W2	[45, 32]
FC_b21	[45, 32]
FC_b22	[32, 1]
Learning Rate	0.0005
Initial Value of ϵ	1.0
Final ϵ Value	0.05
Annealing episodes for ϵ	1100
Batch Size	128
Replay Buffer Size	5000
λ in Q (λ)	0.6

TABLE IV: The learning parameters of MAA2C and COMA which are borrowed from (Foerster et al., 2017)

Parameters	MAA2C	COMA
Actor_FC_1	[15, 64]	[15, 64]
Actor_GRU	[64, 64]	[64, 64]
Actor_FC_2	[64, 5]	[64, 5]
Critic_FC_1	[45, 128]	[85(45+7+3+5*3*2), 128]
Critic_FC_2	[128, 128]	[128, 128]
Critic_FC_3	[128, 1]	[128, 5]
Learning rate of actor	0.0005	0.0005
Learning rate of critic	0.0005	0.0005
Initial Value of ϵ	0.5	0.5
Final ϵ Value	0.02	0.02
Annealing episodes for ϵ	750	750
λ in TD (λ)	/	0.8

For all training experiments, the corresponding parameters are detailed in Table VI, 15,000 episodes are executed in the training phase. The RMSprop optimizer is used consistently, with gradient clipping set to ten to prevent gradient explosion. The discount factor γ is set to 0.99 across all experiments. The target update cycle, representing the frequency of updates to the target Q network in QMIX and the target critic network in the actor-critic algorithm. To ensure reproducibility, the training process starts with a random seed of 2024, incrementing by one for each subsequent episode. To monitor and record the performance of the trained policy, evaluation is conducted every 100 episodes, and the corresponding policy is saved for future reference. The evaluation phase utilizes 40 independent test seeds, completely distinct from those used in training, to prevent overfitting and ensure robust performance assessment. All training experiments are conducted on a single PC with 2.50 GHz Intel(R) Core(TM) i5-12400 CPU and NVIDIA GeForce RTX 3060 GPU.

The weights w_c, w_s, w_{ttc}, w_l, and w_m are set to 1000, 100, 100, 1, and 10, respectively. This ratio is chosen to maintain the dominance of safety-related rewards. The relative magnitudes of them are presented in Table V. As shown, the collision reward and TTC reward have the highest values, followed by the speed reward. The lane change reward and driving task reward have the smallest values. The evaluated vehicles' number N in TTC evaluation is set to 6 leading to the best learning performance.

TABLE V: Magnitude Comparison of Different Kinds of Reward

Weight	**Magnitude**
Collision Reward	1.0×10^3
TTC Reward	1.0×10^2
Speed Reward	1.0×10^1
Lane Change Reward	1.0×10^0
Merging Task Reward	1.0×10^0

TABLE VI: Parameters of Training

Parameters	Value
Total Training Episodes	15000
Optimizer	RMSprop
Gradient Clipping	10
γ	0.99
Target Update Cycle	200
Train Seed	Starting from 2024
Test Cycle	100 Episodes
Test Seed	0, 25, 50, ..., 1000
w_c	1000
w_s	1
w_{ttc}	1
w_l	1
w_m	10
N in TTC	6

A comprehensive analysis of the results demonstrates several notable advantages of QMIX-QLambdaM compared to the baseline algorithms. As illustrated in Figure 5 and Figure 6, QMIX-QLambdaM exhibits superior performance in terms of rewards, maintaining the highest reward learning curve throughout the training process and achieving the highest position in the boxplot distribution. This superiority is quantitatively validated in Table VII, where QMIX-QLambdaM attains the highest mean reward, surpassing the next best performer, QMIX, by a significant margin of 91.

Regarding safety performance, Table VII indicates that QMIX-QLambdaM achieves the lowest collision rate among all algorithms, markedly outperforming the second-best algorithm, QMIX, with a substantial improvement of 74.1%. The collision distribution of the proposed method also demonstrates the most compact and favorable profile.

In terms of efficiency, Figure 8 and Table VII reveal that QMIX-QLambdaM maintains the second-highest average speed, showing only a marginal decrease of 5.0% compared to QMIX, while slightly outperforming MAA2C by 3.0% and significantly exceeding COMA by 33.3%. These results suggest that while QMIX-QLambdaM trades off a modest amount of efficiency to achieve excellent collision avoidance capabilities, the compromise in performance is minimal.

TABLE VII: Comprehensive Comparison of Speed, Collision across Different Algorithms

Indicator	Statistic	QMIX-QLambdaM	QMIX	COMA	MAA2C
Collision (cav/episode)	Mean $\pm$ std	**0.07 $\pm$ 0.09**	0.27 $\pm$ 0.31 (-74.1%)	0.37 $\pm$ 0.23 (-81.1%)	0.35 $\pm$ 0.24 (-80.0%)
Speed (m/s)	Mean $\pm$ std	20.7 $\pm$ 1.9	**21.8 $\pm$ 1.3** (-5.0%)	14.5 $\pm$ 2.7 (+42.8%)	20.1 $\pm$ 4.5 (+3.0%)
Reward	Mean $\pm$ std	**86 $\pm$ 53**	-5 $\pm$ 124 (+91)	-55 $\pm$ 90 (+141)	-32 $\pm$ 106 (+118)

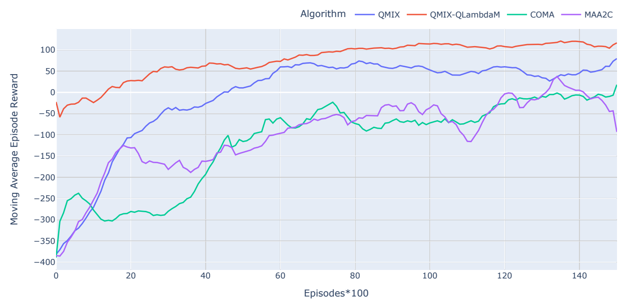


Fig. 5: Moving Average Reward Learning Curve of Different Algorithms (n=9 for better trend visibility)

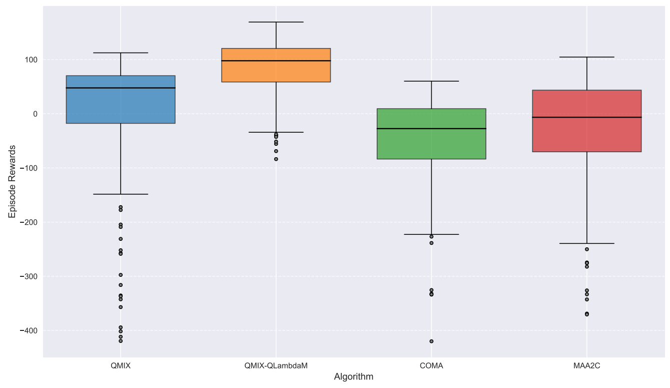


Fig. 6: Statistics of Reward of Different Algorithms of 150 Test Phases Over the Period of Training

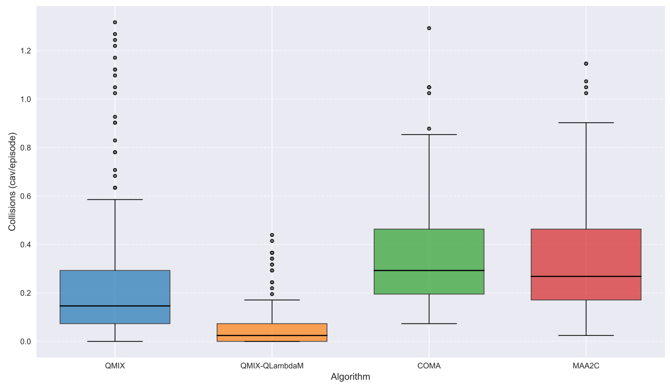


Fig. 7: Statistics of Collision of Different Algorithms of 150 Test Phases Over the Period of Training

C. *Evaluations in Different Traffic Density*

To test the trained policy and evaluate the adaptive ability of our proposed method, we employed the trained policy for cases with different vehicle densities, changing the number of HDVs for each lane. The densities are categorized into three levels: low density which is 9.09 veh/km (3 HDVs, each lane has one HDV), medium density, which is 13.64 veh/km (6 HDVs, each lane has two HDVs) and high density, which is 18.18 veh/km (9 HDVs, each lane has three HDVs). According to the flow-density diagram in (Treiber, 2013), traffic can

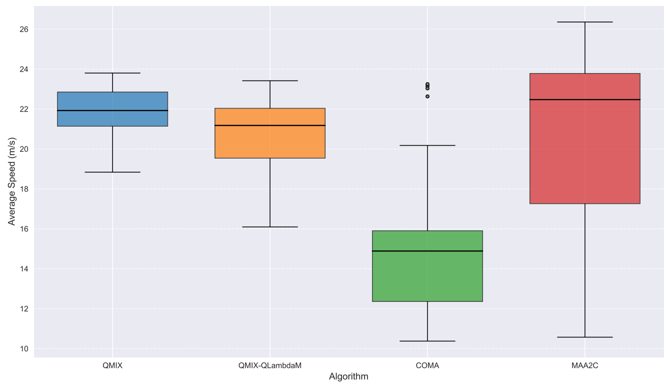


Fig. 8: Statistics of Average Speed of Different Algorithms of 150 Test Phases Over the Period of Training

exist in either free or congested states within a certain density range (18-25 veh/km), with the upper bound of this range still corresponding to free flow conditions. We selected a policy with the best performance (highest reward value) for each algorithm and compared them across these three density levels. The experiments were run with 40 test seeds. The evaluation metrics include total reward, safety reward, efficiency reward, merging task reward, collision rate, and average speed.

The experimental results across varying traffic densities demonstrate distinct performance patterns for each algorithm, as illustrated in Figure 9–Figure 14 and Table VIII. QMIX-QLambdaM demonstrates consistently superior performance, achieving not only the highest mean total rewards but also displaying notably elevated quartile distributions compared to other algorithms, as evidenced by its distinctively higher boxplot position. In low traffic density scenarios, QMIX-QLambdaM's mean total reward surpasses QMIX, COMA, and MAA2C by margins of 84, 130, and 115 respectively. This performance advantage persists in medium density conditions, with margins of 77, 79, and 93 respectively. Similarly, in high-density scenarios, QMIX-QLambdaM maintains its lead with advantages of 109, 123, and 104 respectively.

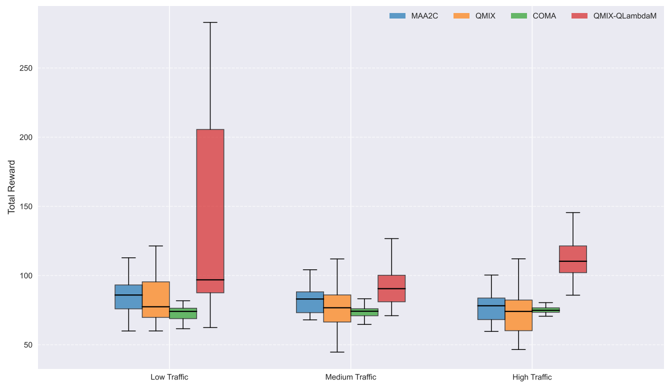


Fig. 9: Statistics of Total Reward for Different Traffic Density

Safety performance, evaluated through collision rates and safety rewards (comprising collision reward and TTC reward), reveals notable distinctions among the algorithms. QMIX-

QLambdaM exhibits exceptional safety characteristics, maintaining zero collision rates and high safety rewards across all density levels, demonstrating its ability to maintain safe gaps and prevent collisions in diverse scenarios which are shown in Figure 10 and Figure 11. In contrast, other algorithms exhibit non-zero collision rates across all traffic densities, moreover, Figure 11 shows that each of the baseline even has two CAVs colliding in the experiments.

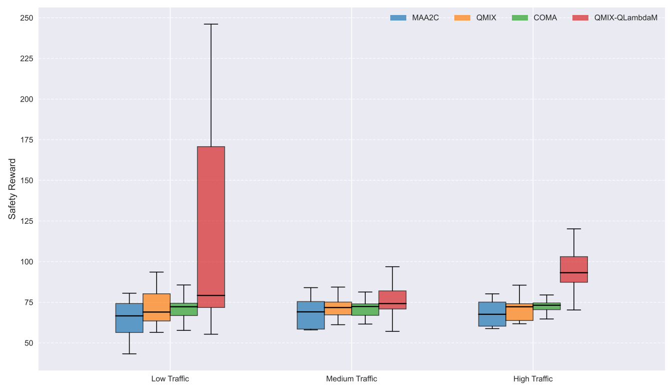


Fig. 10: Statistics of Safety Reward for Different Traffic Density

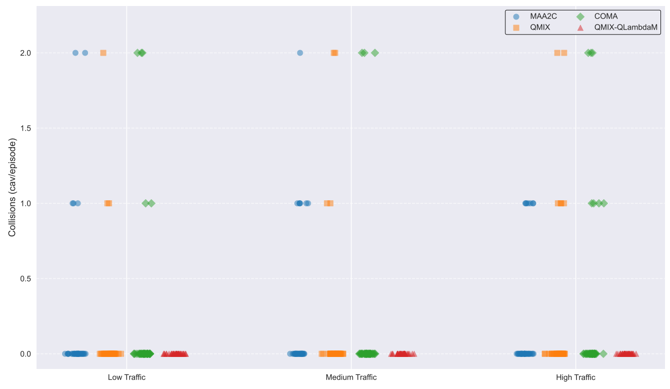


Fig. 11: Statistics of Collisions for Different Traffic Density: the figure is shown with scatter as the boxplot is not readable.

Efficiency performance is evaluated through average speed and efficiency rewards (comprising speed reward and lane change reward) as shown in Figure 13, Figure 12 and Table VIII. The average speed data reveals QMIX-QLambdaM's superior efficiency across all densities. In low density conditions, the proposed method outperforms QMIX, COMA, and MAA2C by 10.8%, 80.3%, and 16% respectively. These advantages persist in medium density (7.6%, 58.2%, and 7.1%) and heavy density scenarios (11.3%, 54.5%, and 11.9%). The efficiency rewards further validate this superior speed performance, with QMIX-QLambdaM achieving the highest values across all densities. Notably, while MAA2C demonstrates a clear advantage in efficiency rewards compared to QMIX despite similar average speeds, this disparity stems from QMIX's excessive lane changes.

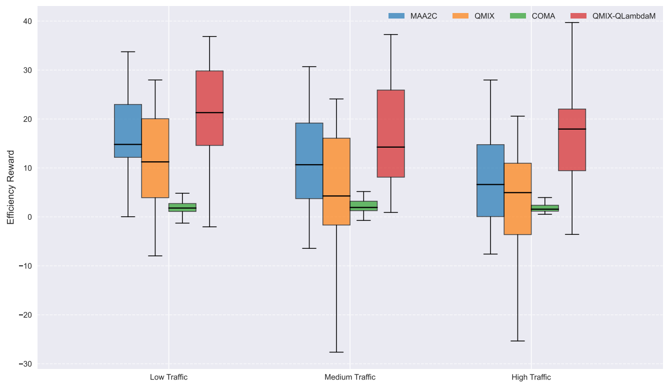


Fig. 12: Statistics of Efficiency Reward for Different Traffic Density

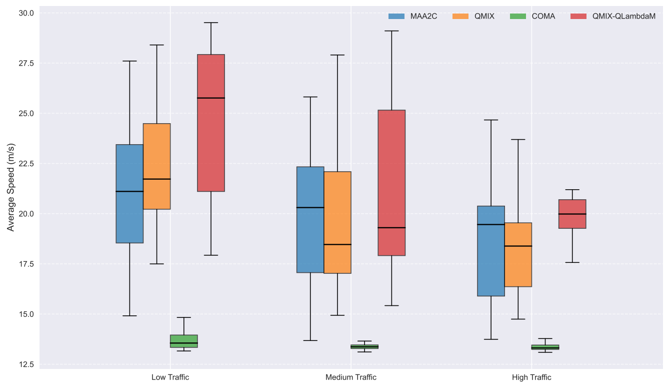


Fig. 13: Statistics of Average Speed for Different Traffic Density

Merging task rewards, which indicate merging efficiency, exhibit varying performance patterns across algorithms and densities. MAA2C achieves optimal mean rewards in low density conditions, while QMIX-QLambdaM excels in medium density scenarios, and COMA demonstrates superior performance in high density situations. Overall, QMIX-QLambdaM demonstrates better merging task rewards than QMIX while performing marginally below the actor-critic methods. However, the high collision rates exhibited by the actor-critic method during merging suggests a trade-off where safety is compromised to achieve merging objectives.

The standard deviations in total rewards provide valuable insights into the consistency of each algorithm's performance. QMIX-QLambdaM exhibits notably lower standard deviations compared to other algorithms, indicating more consistent performance across diverse traffic scenarios. Furthermore, QMIX-QLambdaM demonstrates exceptional stability, particularly in high-density scenarios, where it maintains both superior performance and lower standard deviations in total rewards compared to alternative methods.

D. Decisions and Control for Main-lane and Merging-lane Vehicles

This section demonstrates the strategic control capabilities of our proposed method for main-lane and merging-lane

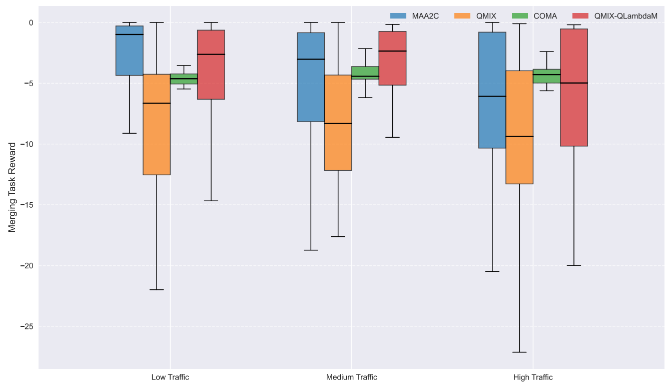


Fig. 14: Statistics of Merging Task Reward for Different Traffic Density

TABLE VIII: Comprehensive Comparison of Total Reward, Safety Reward, Efficiency Reward, Merging Task Reward, Collision, and Average Speed across Different Algorithms in Different Traffic Densities: the proposed method outperforms nearly in every indicator except the merging task reward in low and heavy density

Traffic Density	Indicator	Statistic	QMIX-QLambdaM	QMIX	COMA	MAA2C
Low	Total Reward	Mean ± std	**137 ± 69**	53 ± 144 (+84)	7 ± 203 (+130)	22 ± 184 (+115)
	Safety Reward	Mean ± std	**115 ± 62**	41 ± 139 (+74)	5 ± 201 (+110)	5 ± 179 (+110)
	Efficiency Reward	Mean ± std	**22 ± 11**	12 ± 10 (+10)	2 ± 2 (+20)	17 ± 9 (+5)
	Merging Task Reward	Mean ± std	-5 ± 6	-8 ± 6 (+3)	-5 ± 1 (0)	**-3 ± 5** (-2)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.10 ± 0.37 (-100%)	0.20 ± 0.55 (-100%)	0.17 ± 0.49 (-100%)
	Average Speed (m/s)	Mean ± std	**24.7 ± 3.7**	22.3 ± 2.9 (+10.8%)	13.7 ± 0.4 (+80.3%)	21.3 ± 3.4 (+16.0%)
Medium	Total Reward	Mean ± std	**104 ± 43**	27 ± 172 (+77)	25 ± 186 (+79)	11 ± 169 (+93)
	Safety Reward	Mean ± std	**88 ± 37**	21 ± 170 (+67)	23 ± 185 (+65)	-1 ± 165 (+89)
	Efficiency Reward	Mean ± std	**17 ± 10**	5 ± 12 (+12)	2 ± 2 (+15)	12 ± 10 (+5)
	Merging Task Reward	Mean ± std	**-3 ± 3**	-9 ± 7 (+6)	-4 ± 1 (+1)	-5 ± 6 (+2)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.15 ± 0.47 (-100%)	0.15 ± 0.52 (-100%)	0.20 ± 0.45 (-100%)
	Average Speed (m/s)	Mean ± std	**21.2 ± 4.2**	19.7 ± 3.7 (+7.6%)	13.4 ± 0.2 (+58.2%)	19.8 ± 3.2 (+7.1%)
High	Total Reward	Mean ± std	**116 ± 30**	7 ± 177 (+109)	-7 ± 208 (+123)	12 ± 149 (+104)
	Safety Reward	Mean ± std	**100 ± 26**	4 ± 176 (+96)	-9 ± 207 (+109)	5 ± 144 (+95)
	Efficiency Reward	Mean ± std	**17 ± 10**	3 ± 11 (+14)	2 ± 2 (+15)	7 ± 9 (+10)
	Merging Task Reward	Mean ± std	-6 ± 6	-10 ± 8 (+4)	**-4 ± 1** (-2)	-7 ± 6 (+1)
	Collision (veh/episode)	Mean ± std	**0.00 ± 0.00**	0.20 ± 0.50 (-100%)	0.24 ± 0.58 (-100%)	0.17 ± 0.38 (-100%)
	Average Speed (m/s)	Mean ± std	**20.7 ± 2.6**	18.6 ± 2.9 (+11.3%)	13.4 ± 0.2 (+54.5%)	18.5 ± 3.00 (+11.9%)

vehicles, comparing it with the best-performing baseline to highlight its superior performance. To showcase the method's adaptability and robustness, we present data under heavy traffic density conditions. We selected MAA2C as the comparison baseline, being the second-best performer under heavy density. We chose two scenarios (seeds 625 and 900) to elaborate on the strategic control for main-lane and merging-lane vehicles, illustrating the advantages of our proposed method. In these scenarios, the MAA2C policy failed due to inappropriate behavior of both main-lane and merging-lane vehicles, while QMIX-QLambdaM succeeded.

Figure 15 and Figure 16 present a comparison of the two algorithms' low-level control and speed/lane change profiles for the main-lane 1 vehicle in seed 625. Figure 16 reveals that around time step 10, the speed in the MAA2C case drops below 10 m/s, after which the speed curve disappears, indicating a vehicle collision. Figure 17a depicts the scene showing the CAV colliding with the HDV, while Figure 17b illustrates the scene at time step 12. In the speed and acceleration profiles, the QMIX-QLambda curve exhibits frequent speed adjustments before time step 12 to accommodate the speed of the front HDV. It then stabilizes after time step 12, maintaining a safe gap with the preceding vehicle. In contrast, for the MAA2C case, main-lane vehicle 2 changes to the left lane, causing the HDV behind it (the predecessor of main-lane vehicle 1) to decelerate. CAV 1 attempts to slow down to accommodate the HDV's reduced speed, as evident in Figure 15 and Figure 16. However, it ultimately fails to avoid collision and even accelerates dramatically before impact. Additionally, the positive and negative peaks in acceleration profiles indicate the activation of `acceleration` and `deceleration` actions respectively. As shown in Figure 15, prior to time step 10, QMIX-QLambdaM exhibits a total of 6 peaks, whereas MAA2C shows 7. This lower number of peaks suggests that QMIX-QLambdaM achieves smoother driving behavior.

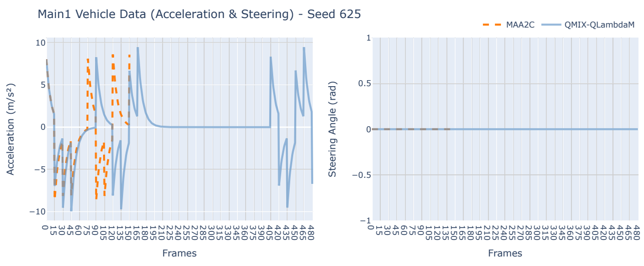


Fig. 15: Low-level control profile of main-lane 1 vehicle of seed 625: the unit is frames as the low level control command varies every frame, 1 time step contains 15 frames

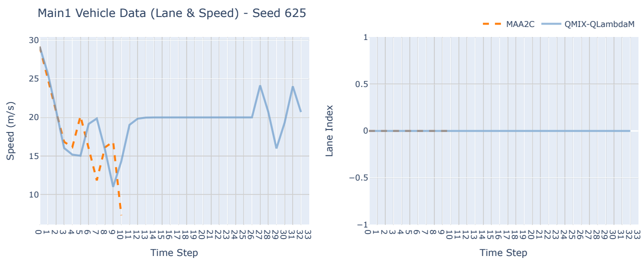
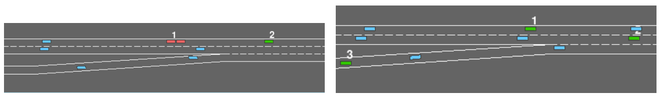


Fig. 16: Speed and lane change profile of main-lane 1 vehicle of seed 625: the speed profile of MAA2C is lower than 10m/s at time step 10 indicating the collision



(a) The scene of collision at time step 10 for MAA2C in seed 625 (b) The scene at time step 12 for QMIX-QLambdaM in seed 625

Fig. 17: Comparison of scenes for MAA2C and QMIX-QLambdaM in seed 625

Figure 19 and Figure 18 illustrate the low-level control commands and speed/lane changes of merging vehicles in seed

900. In Figure 19, lane index 3 indicates the lane prior to the merging vehicle's acceleration lane, lane index 2 indicates the merging lane, lane index 1 indicates the rightmost lane in main-lane, and lane index 0 indicates the leftmost lane in main-lane. The MAA2C curve, which terminates at approximately the 19-time steps, shows a speed below the minimum threshold with a declining trend, indicative of a collision. The lane change profile demonstrates the timing of the merging vehicle's entry into the main lane. As shown in Figure 19 and Figure 18, the MAA2C-controlled merging vehicle begins decelerating at time step 13, attempting to identify a suitable gap. However, at time step 19, it collides with the host HDV in the main lane, as depicted in Figure 20a. In contrast, the QMIX-QLambdaM-controlled merging vehicle successfully identifies a gap at time step 15 (shown in Figure 20b) and maintains a stable speed for the initial 4 time steps post-merge, as evidenced by the acceleration and speed profiles. Furthermore, in the Figure 18, before the time step 19, the total number of the peaks of the acceleration profile of QMIX-QLambdaM (8) is smaller than the value in MAA2C (17). In the steering angle profile, the proposed approach exhibits smaller magnitudes of steering angle changes during lane transitions compared to MAA2C, indicating a more stable driving behavior.

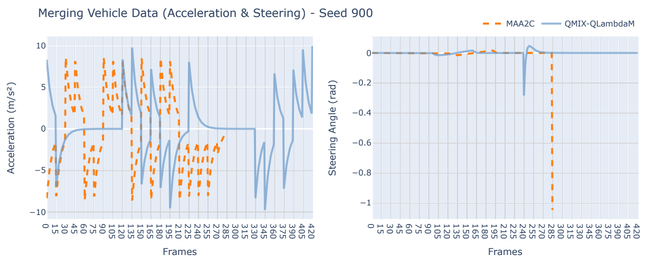


Fig. 18: Low-level control profile of merging vehicle of seed 900

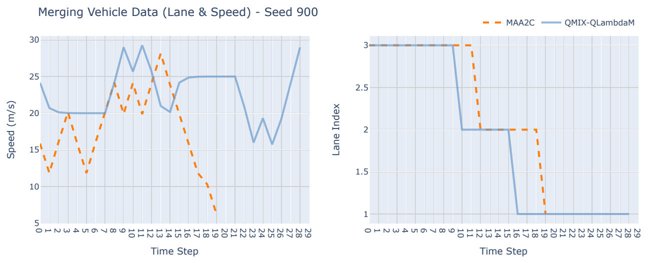
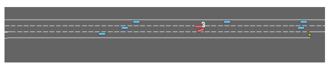


Fig. 19: Speed and lane change profile of merging vehicle of seed 900


(a) The scene of collision at time step 19 for MAA2C in seed 900

(b) The scene at time step 16 for QMIX-QLambdaM in seed 900

Fig. 20: Comparison of scenes for MAA2C and QMIX-QLambdaM in seed 900

In conclusion, these two scenarios (seeds 625 and 900) clearly demonstrate the superior performance of QMIX-QLambdaM over MAA2C in handling complex traffic situations. In the main-lane vehicle scenario (seed 625), QMIX-QLambdaM showcased its ability to make frequent, appropriate speed adjustments to maintain safe distances, ultimately avoiding collision. Conversely, MAA2C's failure to adequately respond to changing traffic conditions led to a collision. In the merging scenario (seed 900), QMIX-QLambdaM demonstrated superior gap identification and smoother lane change execution, resulting in a successful merge without incident. MAA2C, however, failed to safely navigate the merge, resulting in a collision. These results underscore QMIX-QLambdaM's enhanced adaptability, safety, and stability in both longitudinal and lateral vehicle control.

V. Conclusion and Future Works

This paper presented QMIX-QLambdaM, an improved QMIX algorithm incorporating Q(λ) returns and action masking, to address the challenge of on-ramp merging for CAVs in mixed traffic environments. The proposed approach was formulated as a CTDE MARL problem, capable of handling dynamic scenarios with both CAVs and HDVs. In the learning curve comparison, QMIX-QLambdaM demonstrated an advantage in terms of credit assignment compared to other CTDE baselines for the formulated problem. Both learning curve analyses and comprehensive experiments showed that QMIX-QLambdaM consistently outperformed state-of-the-art algorithms, including QMIX, MAA2C, and COMA, across various performance metrics. The proposed method exhibited superior adaptability across different traffic densities, maintaining high performance in terms of safety, efficiency, and overall rewards. Furthermore, case studies illustrated QMIX-QLambdaM's ability to generate effective strategic control for both main-lane and merging-lane vehicles, showcasing smoother driving behavior and better collision avoidance compared to baseline methods.

Future research directions may include extending the approach to more diverse and realistic traffic scenarios and settings, while also investigating different altruistic and egoistic agents with various rewards related to social impacts. Incorporating more sophisticated models of human driving behavior is another crucial avenue, particularly focusing on uncertain behavior models and changeable behaviors as highlighted by (Toghi et al., 2021). Additionally, exploring transfer learning techniques presents an interesting direction for enhancing scalability. The current RNN-based learning structure faces limitations due to its fixed input dimension, which poses challenges for scalability. Recent studies, such as the work by (Hu et al., 2021), suggest that replacing RNNs with transformer architectures could effectively address this issue, potentially improving the model's adaptability to varying numbers of agents and environmental complexities.

References

Athans, M. (1969). A unified approach to the vehicle-merging problem. *Transportation Research*, 3(1):123–133.

Bouton, M., Nakhaei, A., Fujimura, K., and Kochenderfer, M. J. (2019). Cooperation-aware reinforcement learning for merging in dense traffic. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3441–3447.

Cao, W., Mukai, M., Kawabe, T., Nishira, H., and Fujiki, N. (2015). Cooperative vehicle path generation during merging using model predictive control with real-time optimization. *Control Engineering Practice*, 34:98–105.

Chen, D., Hajidavalloo, M. R., Li, Z., Chen, K., Wang, Y., Jiang, L., and Wang, Y. (2023). Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):11623–11638.

Dey, K. C., Rayamajhi, A., Chowdhury, M., Bhavsar, P., and Martin, J. (2016). Vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network – performance evaluation. *Transportation Research Part C: Emerging Technologies*, 68:168–184.

Du, Y., Leibo, J. Z., Islam, U., Willis, R., and Sunehag, P. (2023). A review of cooperation in multi-agent learning. *CoRR*, abs/2312.05162.

Dutch Design Guidelines (2024). Dutch design guidelines. https://standaarden.rws.nl/index.html. Accessed: [Insert access date here].

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2017). Counterfactual multi-agent policy gradients.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797.

Hu, J., Jiang, S., Harding, S. A., Wu, H., and wei Liao, S. (2023). Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning.

Hu, S., Zhu, F., Chang, X., and Liang, X. (2021). Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers.

Hu, Y., Nakhaei, A., Tomizuka, M., and Fujimura, K. (2019). Interaction-aware decision making with adaptive strategies under merging scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–158.

Irshayyid, A., Chen, J., and Xiong, G. (2024). A review on reinforcement learning-based highway autonomous vehicle control. *Green Energy and Intelligent Transportation*, 3(4):100156.

Jiao, Y. (2023). A fast calculation of two-dimensional Time-to-Collision.

Kesting, A., Treiber, M., and Helbing, D. (2007). General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94.

Kruber, F., Wurst, J., Chakraborty, S., and Botsch, M. (2019). Highway traffic data: macroscopic, microscopic and criticality analysis for capturing relevant traffic scenarios and traffic modeling based on the highd data set.

Leurent, E. (2018). An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env.

Marinescu, D., Čurn, J., Bouroche, M., and Cahill, V. (2012). On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 900–906.

Min, H., Fang, Y., Wu, X., Wu, G., and Zhao, X. (2021). On-ramp merging strategy for connected and automated vehicles based on complete information static game. *Journal of Traffic and Transportation Engineering (English Edition)*, 8(4):582–595.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Peng, J. and Williams, R. J. (1994). Incremental multi-step q-learning. In Cohen, W. W. and Hirsh, H., editors, *Machine Learning Proceedings 1994*, pages 226–232. Morgan Kaufmann, San Francisco (CA).

Polack, P., Altché, F., Novel, B., and de La Fortelle, A. (2017). The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? pages 812–818.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. (2020a). Weighted qmix: Expanding monotonic value function factorisation. *ArXiv*, abs/2006.10800.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning.

Rashid, T., Samvelyan, M., Witt, C. S. D., Farquhar, G., Foerster, J. N., and Whiteson, S. (2020b). Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21:178:1–178:51.

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. (2019). The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 2186–2188, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Schmidt, G. K. and Posch, B. (1983). A two-layer control scheme for merging of automated vehicles. In *The 22nd IEEE Conference on Decision and Control*, pages 495–500.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. (2019). Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *ArXiv*, abs/1905.05408.

Sumanth Nakka, S. K., Chalaki, B., and Malikopoulos, A. A. (2022). A multi-agent deep reinforcement learning coordination framework for connected and automated vehicles at merging roadways. In *2022 American Control Conference (ACC)*, pages 3297–3302.

Sun, Z., Huang, T., and Zhang, P. (2020). Cooperative decision-making for mixed traffic: A ramp merging example. *Transportation research part C: emerging technologies*, 120:102764.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V. F., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Adaptive Agents and Multi-Agent Systems*.

Thiemann, C., Treiber, M., and Kesting, A. (2008). Estimating acceleration and lane-changing dynamics from next generation simulation trajectory data. *Transportation Research Record*, 2088(1):90–101.

Toghi, B., Valiente, R., Sadigh, D., Pedarsani, R., and Fallah, Y. P. (2021). Altruistic maneuver planning for cooperative autonomous vehicles using multi-agent advantage actor-critic.

Treiber, M. (2013). *Traffic Flow Dynamics*.

Treiber, M., Hennecke, A., and Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824.

Triest, S., Villaflor, A., and Dolan, J. M. (2020). Learning highway ramp merging via reinforcement learning with temporally-extended actions. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1595–1600.

Uno, A., Sakaguchi, T., and Tsugawa, S. (1999). A merging control algorithm based on inter-vehicle communication. In *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*, pages 783–787.

US Department of Transportation (2018). A policy on geometric design of highways and streets.

Valiente, R., Toghi, B., Pedarsani, R., and Fallah, Y. P. (2022). Robustness and adaptability of reinforcement learning based cooperative autonomous driving in mixed-autonomy traffic.

Wang, P. and Chan, C.-Y. (2017). Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE.

Yu, C., Wang, X., Xu, X., Zhang, M., Ge, H., Ren, J., Sun, L., Chen, B., and Tan, G. (2020). Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs. *IEEE Transactions on Intelligent Transportation Systems*, 21(2):735–748.

Zong, S. (2019). How connected autonomous vehicles would affect our world? —— a literature review on the impacts of cav on road capacity, environment and public attitude. *MATEC Web of Conferences*, 296:01007.

B Action Masking Mechanism Supplementary

Algorithm 3 UnsafeVehicle

```
function UnsafeVehicle($v$, $v_{surr}$)
    $d = |v_{surr}[X] - v[x]|$
    if $d \leq TTC_{safe} \times (v_{max} - v_{min})$ then
        return $v_{surr}$
    else
        return $None$
    end if
end function
```

Algorithm 4 1DTTC

```
function 1DTTC($ego$, $surr$)
    if $ego.position_x \geq surr.position_x$ then
        $front \leftarrow ego$
        $rear \leftarrow surr$
    else
        $front \leftarrow surr$
        $rear \leftarrow ego$
    end if
    $\Delta x \leftarrow front.position_x - rear.position_x$
    $\Delta v_x \leftarrow rear.speed - front.speed$
    $relative\_position\_x \leftarrow \Delta x$
    $combined\_length \leftarrow (front.LENGTH + rear.LENGTH)/2$
    if $\Delta v_x = 0$ then
        if $relative\_position\_x > combined\_length$ then
            return $\infty$
        else
            return $-1$
        end if
    end if
    if $\Delta v_x > 0$ then
        $ttc \leftarrow (relative\_position\_x - combined\_length)/\Delta v_x$
        return $ttc$
    end if
    if $\Delta v_x < 0$ then
        return $\infty$
    end if
end function
```

Algorithm 5 AccelerateMask

```
function accelerate_mask($ego$, $v_f$, $available\_actions$, $t$)
    if $acceleration$ is available or $idle$ is available then
        for frame in $t$ do
            if $acceleration$ is available then
                $ego$ accelerates
            else
                $ego$ idles
            end if
            if $v_f$ is a controlled vehicle then
                $v_f$ decelerates
            else if $v_f$ is human vehicle then
                $v_f$ follows IDM policy
            end if
            $TTC_f \leftarrow$ 1DTTC($ego$, $v_f$)
            if $TTC_f \leq TTC_{safe}$ then
                if any of $deceleration, turning\ left, turning\ right$ is available then
                    if $idle$ is available then
                        $idle$ is masked out
                    end if
                    if $acceleration$ is available then
                        $acceleration$ is masked out
                    end if
                else
                    $acceleration$ is masked out
                end if
                break
            end if
        end for
    end if
    return $available\_actions$
end function
```

Algorithm 6 DecelerateMask

```
function accelerate_mask($ego$, $v_r$, $available\_actions$, $t$)
    if $deceleration$ is available or $idle$ is available then
        for frame in $t$ do
            if $deceleration$ is available then
                $ego$ decelerates
            else
                $ego$ idles
            end if
            if $v_r$ is a controlled vehicle then
                $v_r$ accelerates
            else if $v_r$ is human vehicle then
                $v_r$ follows IDM policy
            end if
            $TTC_r \leftarrow$ 1DTTC($ego$, $v_r$)
            if $TTC_r \leq TTC_{safe}$ then
                if any of $acceleration, turning\ left, turning\ right$ is available then
                    if $idle$ is available then
                        $idle$ is masked out
                    end if
                    if $deceleration$ is available then
                        $deceleration$ is masked out
                    end if
                else
                    $deceleration$ is masked out
                end if
                break
            end if
        end for
    end if
    return $available\_actions$
end function
```

Algorithm 7 LeftMask

```
function LeftMask($ego, v_{lf}, v_{lr}, available\_actions, t$)
    if $turning\ left$ is available then
        for tick in $t$ do
            $ego$ turn left
            if $v_{lf} \neq None$ and $v_{lr} \neq None$ then
                $collision$ ← PredictTrajectoryLateral($ego, v_{lf}$)
                if $collision$ then
                    $turning\ left$ is masked out
                    break
                end if
                $collision$ ← PredictTrajectoryLateral($ego, v_{lr}$)
                if $collision$ then
                    $turning\ left$ is masked out
                    break
                end if
            end if
            if $v_{lf} \neq None$ && $v_{lr} = None$ then
                $collision$ ← PredictTrajectoryLateral($ego, v_{lf}$)
                if $collision$ then
                    $turning\ left$ is masked out
                    break
                end if
            end if
            if $v_{lf} = None$ && $v_{lr} \neq None$ then
                $collision$ ← PredictTrajectoryLateral($ego, v_{lr}$)
                if $collision$ then
                    $turning\ left$ is masked out
                    break
                end if
            end if
        end for
    end if
    return $available\_actions$
end function
```

Algorithm 8 RightMask

```
function RightMask($ego, v_{rf}, v_{rr}, available\_actions, t$)
    if $turning\ right$ is available then
        for tick in $t$ do
            $ego$ turn right
            if $v_{rf} \neq None$ and $v_{rr} \neq None$ then
                $collision \leftarrow$ PredictTrajectoryLateral($ego, v_{rf}$)
                if $collision$ then
                    $turning\ right$ is masked out
                    break
                end if
                $collision \leftarrow$ PredictTrajectoryLateral($ego, v_{rr}$)
                if $collision$ then
                    $turning\ right$ is masked out
                    break
                end if
            end if
            if $v_{rf} \neq None$ && $v_{rr} = None$ then
                $collision \leftarrow$ PredictTrajectoryLateral($ego, v_{rf}$)
                if $collision$ then
                    $turning\ right$ is masked out
                    break
                end if
            end if
            if $v_{rf} = None$ && $v_{rr} \neq None$ then
                $collision \leftarrow$ PredictTrajectoryLateral($ego\ vehicle, v_{rr}$)
                if $collision$ then
                    $turning\ right$ is masked out
                    break
                end if
            end if
        end for
    end if
    return $available\_actions$
end function
```

Algorithm 9 PredictTrajectoryLateral

```
function PredictTrajectoryLateral($v, v_{lf}, v_{lr}, v_{rf}, v_{rr}$)
    if $v_{lf}$ is controlled vehicle then
        if $v_{lf}[lane\_number] = v[lane\_number] - 1$ then
            $v_{lf}$ decelerate
        else
            $v_{lf}$ turn right
        end if
        if $v_{lf}$ is human-driven vehicle then
            $v_{lf}$ follows IDM
        end if
        Check the $collision$ for $v$ and $v_{lf}$
        return $collision$                                   ▷ Same logic for $v_{lr}, v_{rf}, v_{rr}$
    end if
end function
```

Bibliography

Antoniotti, M., A. Desphande, and A. Girault (1997). “Microsimulation analysis of multiple merge junctions under autonomous AHS operation”. In: *Proceedings of Conference on Intelligent Transportation Systems*, pp. 147–152. doi: 10.1109/ITSC.1997.660466.

Athans, Michael (1969). “A unified approach to the vehicle-merging problem”. In: *Transportation Research* 3.1, pp. 123–133. issn: 0041-1647. doi: https://doi.org/10.1016/0041-1647(69)90109-9.

Bouton, Maxime et al. (2019). “Cooperation-Aware Reinforcement Learning for Merging in Dense Traffic”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3441–3447. doi: 10.1109/ITSC.2019.8916924.

Cao, Wenjing et al. (2015). “Cooperative vehicle path generation during merging using model predictive control with real-time optimization”. In: *Control Engineering Practice* 34, pp. 98–105. issn: 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2014.10.005.

Chen, Dong et al. (2023). “Deep Multi-Agent Reinforcement Learning for Highway On-Ramp Merging in Mixed Traffic”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.11, pp. 11623–11638. doi: 10.1109/TITS.2023.3285442.

Deichmann, Johannes (2023). *Autonomous driving’s future: Convenient and connected*. McKinsey.

Dey, Kakan Chandra et al. (2016). “Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network – Performance evaluation”. In: *Transportation Research Part C: Emerging Technologies* 68, pp. 168–184. issn: 0968-090X. doi: https://doi.org/10.1016/j.trc.2016.03.008.

Du, Yali et al. (2023). “A Review of Cooperation in Multi-agent Learning”. In: *CoRR* abs/2312.05162. url: https://doi.org/10.48550/arXiv.2312.05162.

Duret, Aurelien, Meng Wang, and Andres Ladino (2020). “A hierarchical approach for splitting truck platoons near network discontinuities”. In: *Transportation Research Part B: Methodological* 132. 23rd International Symposium on Transportation and Traffic Theory (ISTTT 23), pp. 285–302. issn: 0191-2615. doi: https://doi.org/10.1016/j.trb.2019.04.006.

Dutch Design Guidelines (2024). *Dutch Design Guidelines*. https://standaarden.rws.nl/index.html. Accessed: [Insert access date here].

Foerster, Jakob et al. (2017). *Counterfactual Multi-Agent Policy Gradients*. arXiv: 1705.08926 [cs.AI]. url: https://arxiv.org/abs/1705.08926.

Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor (Oct. 2019). “A survey and critique of multiagent deep reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.6, pp. 750–797. issn: 1573-7454. doi: 10.1007/s10458-019-09421-1.

Hu, Jian et al. (2023). *Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning*. arXiv: 2102.03479 [cs.LG]. url: https://arxiv.org/abs/2102.03479.

Hu, Siyi et al. (2021). *UPDeT: Universal Multi-agent Reinforcement Learning via Policy Decoupling with Transformers*. arXiv: 2101.08001 [cs.LG]. url: https://arxiv.org/abs/2101.08001.

Hu, Yeping et al. (2019). “Interaction-aware Decision Making with Adaptive Strategies under Merging Scenarios”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 151–158. doi: 10.1109/IROS40897.2019.8968478.

Irshayyid, Ali, Jun Chen, and Guojiang Xiong (2024). “A review on reinforcement learning-based highway autonomous vehicle control”. In: *Green Energy and Intelligent Transportation* 3.4, p. 100156. issn: 2773-1537. doi: https://doi.org/10.1016/j.geits.2024.100156.

Jiao, Yiru (Mar. 2023). *A fast calculation of two-dimensional Time-to-Collision*. url: https://github.com/Yiru-Jiao/Two-Dimensional-Time-To-Collision.

Kesting, Arne, Martin Treiber, and Dirk Helbing (2007). “General Lane-Changing Model MOBIL for Car-Following Models”. In: *Transportation Research Record* 1999.1, pp. 86–94. doi: 10.3141/1999-10. eprint: https://doi.org/10.3141/1999-10.

Kruber, Friedrich et al. (2019). *Highway traffic data: macroscopic, microscopic and criticality analysis for capturing relevant traffic scenarios and traffic modeling based on the highD data set*. arXiv: 1903.04249.

Leurent, Edouard (2018). *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env.

Marinescu, Dan et al. (2012). "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 900–906. doi: 10.1109/ITSC.2012.6338779.

Min, Haigen et al. (2021). "On-ramp merging strategy for connected and automated vehicles based on complete information static game". In: *Journal of Traffic and Transportation Engineering (English Edition)* 8.4, pp. 582–595. issn: 2095-7564. doi: https://doi.org/10.1016/j.jtte.2021.07.003.

Mnih, Volodymyr et al. (Feb. 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. issn: 00280836. url: http://dx.doi.org/10.1038/nature14236.

Polack, Philip et al. (June 2017). "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: pp. 812–818. doi: 10.1109/IVS.2017.7995816.

Rashid, Tabish, Gregory Farquhar, et al. (2020). "Weighted QMIX: Expanding Monotonic Value Function Factorisation". In: *ArXiv* abs/2006.10800. url: https://api.semanticscholar.org/CorpusID:225053994.

Rashid, Tabish, Mikayel Samvelyan, C. S. D. Witt, et al. (2020). "Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". In: *J. Mach. Learn. Res.* 21, 178:1–178:51. url: https://api.semanticscholar.org/CorpusID:213176860.

Rashid, Tabish, Mikayel Samvelyan, Christian Schroeder de Witt, et al. (2018). *QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning*. arXiv: 1803.11485 [cs.LG]. url: https://arxiv.org/abs/1803.11485.

Rios-Torres, Jackeline and Andreas A. Malikopoulos (2017). "A Survey on the Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5, pp. 1066–1077. doi: 10.1109/TITS.2016.2600504.

Samvelyan, Mikayel et al. (2019). "The StarCraft Multi-Agent Challenge". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Montreal QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, pp. 2186–2188. isbn: 9781450363099.

Schmidt, Gunther K. and Bernd Posch (1983). "A two-layer control scheme for merging of automated vehicles". In: *The 22nd IEEE Conference on Decision and Control*, pp. 495–500. doi: 10.1109/CDC.1983.269891.

Son, Kyunghwan et al. (2019). "QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning". In: *ArXiv* abs/1905.05408. url: https://api.semanticscholar.org/CorpusID:153312464.

Sumanth Nakka, Sai Krishna, Behdad Chalaki, and Andreas A. Malikopoulos (2022). "A Multi-Agent Deep Reinforcement Learning Coordination Framework for Connected and Automated Vehicles at Merging Roadways". In: *2022 American Control Conference (ACC)*, pp. 3297–3302. doi: 10.23919/ACC53348.2022.9867314.

Sun, Zhanbo, Tianyu Huang, and Peitong Zhang (2020). "Cooperative decision-making for mixed traffic: A ramp merging example". In: *Transportation research part C: emerging technologies* 120, p. 102764.

Sunehag, Peter et al. (2018). "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward". In: *Adaptive Agents and Multi-Agent Systems*. url: https://api.semanticscholar.org/CorpusID:260498115.

Thiemann, Christian, Martin Treiber, and Arne Kesting (2008). "Estimating Acceleration and Lane-Changing Dynamics from Next Generation Simulation Trajectory Data". In: *Transportation Research Record* 2088.1, pp. 90–101. doi: 10.3141/2088-10. eprint: https://doi.org/10.3141/2088-10.

Toghi, Behrad et al. (2021). *Altruistic Maneuver Planning for Cooperative Autonomous Vehicles Using Multi-agent Advantage Actor-Critic*. arXiv: 2107.05664 [cs.RO]. url: https://arxiv.org/abs/2107.05664.

Treiber, Martin (Jan. 2013). *Traffic Flow Dynamics*. isbn: 978-3-642-32459-8. doi: 10.1007/978-3-642-32460-4.

Treiber, Martin, Ansgar Hennecke, and Dirk Helbing (Aug. 2000). “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical Review E* 62.2, pp. 1805–1824. issn: 1095-3787. doi: 10.1103/physreve.62.1805.

Triest, Samuel, Adam Villaflor, and John M. Dolan (2020). “Learning Highway Ramp Merging Via Reinforcement Learning with Temporally-Extended Actions”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1595–1600. doi: 10.1109/IV47402.2020.9304841.

Uno, A., T. Sakaguchi, and S. Tsugawa (1999). “A merging control algorithm based on inter-vehicle communication”. In: *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*, pp. 783–787. doi: 10.1109/ITSC.1999.821160.

US Department of Transportation (2018). *A Policy on Geometric Design of Highways and Streets*. url: https://highways.dot.gov/safety/pedestrian-bicyclist/safety-tools/417-policy-geometric-design-highways-and-streets-6th (visited on 09/12/2024).

Valiente, Rodolfo et al. (2022). *Robustness and Adaptability of Reinforcement Learning based Cooperative Autonomous Driving in Mixed-autonomy Traffic*. arXiv: 2202.00881 [cs.RO]. url: https://arxiv.org/abs/2202.00881.

Wang, Pin and Ching-Yao Chan (Oct. 2017). “Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. doi: 10.1109/itsc.2017.8317735.

Yu, Chao et al. (2020). “Distributed Multiagent Coordinated Learning for Autonomous Driving in Highways Based on Dynamic Coordination Graphs”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.2, pp. 735–748. doi: 10.1109/TITS.2019.2893683.

Zong, Shuya (Jan. 2019). “How Connected Autonomous Vehicles Would Affect Our World? —— A Literature Review on the Impacts of CAV on Road Capacity, Environment and Public Attitude”. In: *MATEC Web of Conferences* 296, p. 01007. doi: 10.1051/matecconf/201929601007.