World Model based Multi-agent Proximal Policy Optimization Framework for Multi-agent Pathfinding

by

Jaehoon Chung
B.Eng., Korea University, 2019
M.Eng., Korea University, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

World Model based Multi-agent Proximal Policy Optimization Framework for Multi-agent Pathfinding

by

Jaehoon Chung
B.Eng., Korea University, 2019
M.Eng., Korea University, 2021

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor
(Department of Mechanical Engineering)

Dr. Hong-Chuan Yang, Supervisory Committee Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Multi-agent pathfinding plays a crucial role in various robot applications. Recently, deep reinforcement learning methods have been adopted to solve large-scale planning problems in a decentralized manner. Nonetheless, such approaches pose challenges such as non-stationarity and partial observability. This thesis addresses these challenges by introducing a centralized communication block into a multi-agent proximal policy optimization framework. The evaluation is conducted in a simulation based environment, featuring continuous state and action spaces. The simulator consists of a vectorized 2D physics engine where agents are bound by the laws of physics.

Within the framework, a World model is utilized to extract and abstract representation features from the global map, leveraging the global context to enhance the training process. This approach involves decoupling the feature extractor from the agent training process, enabling a more accurate representation of the global state that remains unbiased by the actions of the agents. Furthermore, the modularized approach offers the flexibility to replace the representation model with another model or modify tasks within the global map without the retraining of the agents.

The empirical study demonstrates the effectiveness of the proposed approach by comparing three proximal policy optimization-based multi-agent pathfinding frameworks. The results indicate that utilizing an autoencoder-based state representation model as the centralized communication model sufficiently provides the global context. Additionally, introducing centralized communication block improves performance and the generalization capability of agent policies.

Table of Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Najjaran. During times when I lacked confidence in myself, he believed in me and held expectations for my progress. Even when there were no significant advancements, he patiently waited and consistently provided invaluable support. His mentorship and the research environment he fostered for students have been instrumental in allowing me to successfully complete this journey.

I also wish to extend my heartfelt thanks to Dr. Younes and Dr. Fayyad, who offered precious guidance when I was feeling lost. Their direction enabled me to learn and gain so much throughout this process. Additionally, my colleagues at the ACIS lab have been a great source of inspiration. The time spent with them will remain a cherished memory for a long time.

Lastly, but most importantly, I want to express my deepest gratitude to my beloved wife. During moments of adversity, her mere presence provided the strength and foundation for me to grow even more. Thanks to her constant love and unwavering belief in me. I also giva all glory to God, who has guided me so far and will continue to lead my journey.

Chapter 1

Introduction

This chapter introduces the comprehensive overview, beginning with the demand and motivation for solving Multi-agent Pathfinding (MAPF) problem in industrial settings where multiple robots interact. It explores the challenges and constraints confronted in realistic applications and highlights the research objectives that arise from these limitations and presents the contribution of this thesis with its novel approach to address them. In addition, The chapter provides the outline of the thesis, providing a roadmap for the subsequent chapters.

1.1 Motivation

In recent years, the demand for artificial intelligence (AI) systems that utilize collaborative multi-agent systems has grown substantially across various industries. The market for autonomous multi-agents has rapidly expanded, growing from $ 3.2 billion in 2022, to $ 4.8 billion in 2023, and is projected to reach $ 28.5 billion by 2028, exhibiting a compound annual growth rate of 36.5%. These settings typically involve moving physical robots that interact with each other to achieve cooperative or competitive objectives. It can be found in diverse settings, including assembly [36], warehouse [122],

evacuation [47], formation [3, 69], localization [24], micro-droplet manipulation [27], object transportation [75], search-and-rescue [45], and air-traffic management [90]. Moving multiple robots in these scenarios is significantly more challenging than handling a single robot. Each movement of an agent can cause interference with other agents, leading to potential collisions or more complex conditions for achieving their objectives. The complexity increases even more drastically as the number of agents grows up. To successfully navigate these challenges and ensure the agents to complete their missions, effective team navigation is essential to prevent collisions and avoid obstructing other agents goals.

The MAPF problem aims to derive a team navigation solution that involves a group of agents to plan and follow paths from their starting positions to target destinations without colliding. Consequently, solving the MAPF problem typically becomes the first step in developingthe multi-robot frameworks. However, the difficulty of solving this problem can range from easy to extremely challenging, depending on the conditions, assumptions, and constraints. For instance, when agents can completely observe all other agents and the entire environment, they are able to create safe and optimized paths. However, as the number of agents increases or the global map size expands, the computational cost exponentially increases, posing limitations for real-time applications. Moreover, it is typically impractical for real-world robots to function under fully observable conditions. In contrast, partially-observable environments, where robots only have access to local observations, enable distributed computation by decentralizing the MAPF solution, making them more realistic for industrial applications.

One of the primary challenges in solving MAPF in decentralized manner is that these approaches inevitably bring about non-stationarity issues and limitations in accessing complete environmental information due to partial observability. Non-

stationarity arises from the dynamic nature of the environment, where movements from other agents constantly change the environment. Hence, the predictions based on previous experiences lie less reliable over time. On the other hand, partial observability refers to each agent's limited observation range, restricting direct access to all relevant environmental states necessary for decision-making.

One of the efforts to mitigate non-stationarity in MAPF is fostering communication among agents. DHC [58] achieved this by integrating graph convolutional communication blocks and guided Reinforcement Learning (RL), successfully enhancing the stationary. Their subsequent work further improved communication by minimizing redundant information in broadcast communication, ensuring agents access only pertinent data for decision-making. The authors of [10] addressed non-stationarity in MAPF without explicit communication but by capturing implicit collaborative information between agents. Similarly, PICO [51] integrated implicit planning priorities with a learning-based communication approach, showcasing the effectiveness.

Another challenge in MAPF posed by partial observability has been addressed through the Centralized Training and Decentralized Execution (CTDE) scheme. In this approach, agents access the environmental state representation and team rewards during the training phase, whereas the learned policies rely solely on each agent's local observation during the execution phase. This framework reduces the likelihood of agents making undesirable selfish behaviors, fostering agents to align with team objectives or collaborative rewards. However, since the full state of the environment is normally not known in most industrial settings, the environmental state is represented by aggregation of local observations from the agents. Hence, the effectiveness of CTDE scheme depends on the accurate representation of the environmental state.

The challenges of decentralized MAPF leaves several research opportunities to reduce uncertainty or to present more robust and reliable solutions that strike a balance

with partial observability. Moreover, the problems can become even more challenging under realistic conditions and features of partial observability in terms of industrial applications. For instance, although the problem would be easier if the agents had a longer range of local observations, this would increase equipment costs. Additionally, agents may reduce the environmental uncertainty if they could distinguish other agents from obstacles, but this would require an additional object detection module before processing the MAPF planner. The primary motivation behind this study is to develop an AI-based MAPF planner under the limited conditions and features of partial observability expected when equipping agents with basic sensors. This thesis proposes a novel AI framework to reduce the uncertainty faced by agents and improve the robustness of the MAPF solution.

1.2 Contributions

This thesis investigates how Deep Reinforcement Learning (DRL) based decentralized MAPF planners can take advantage of global context to mitigate partial observability and improve the policies. MAPF research community has been studied improving DRL solutions for more complex environment with larger scale and has achieved significant progresses. Nevertheless, most of the works have been focusing on grid map environments where the state space and the action space are discrete, restricting the flexibility of dynamics and movements of agents. Furthermore, these works do not take into account the actual physical dynamics of the agents, which may lead to challenges in real-world applications.

From a practical application perspective, this thesis focuses on a continuous environment where agents operate under the laws of physics. Additionally, most of other partially observable environments where agents access higher level of local

observation, including information of obstacle locations, other agents' locations and goal locations. This thesis assumes that agents should only rely on LIDAR sensory data where only the distance toward objects can be read without noticing the identity of them. In this regard, the primary contributions of this thesis can be summarized as follows:

- Developed a novel framework for extracting abstract features that capture the sailent features of the global context from global image map. It was demonstrated that the abstracted global context contains enough information to assist agents by experimental studies.
- Proposed decoupling the feature extractor network of the global map from the agent training framework. This allows updating the feature extractor or modifying the task without retraining the agent networks.
- Demonstrated the effectiveness of the proposed approach through a wide range of experiments, including a comparison with two other Proximal Policy Optimization (PPO) frameworks in MAPF.

1.3 Thesis Outline

This thesis is structured into six Chapters, each contributing to different facets of MAPF. The overview of the Chapters is provided as follows:

Chapter 2 provides an overview of important concepts and theories that formulates the basis of the subject in this thesis. It provides an extensive literature review regarding MAPF planners, problem formulation and the objective of MAPF problem, the fundamental framework of MAPF to be applicable for DRL,

mathematical formulation and theories of DRL approaches, and discusses recent progresses of DRL-based approaches.

Chapter 3 provides a details of the environment and data collection. The environment introduced in this Chapter is utilized as the experimental, training, and testing setup of the framework proposed in this thesis. The collected data is used for training process in Chapter 4 and the experimental setup is adopted as the configuration in Chapter 4 and Chapter 5.

Chapter 4 presents a novel network model that abstracts global image map and extracts essential features of global context. The detailed training architecture is provided and it discusses ablation study on skip connections and training slope analysis. The trained model is adopted as the centralized communication block in Chapter 5.

Chapter 5 proposes World Model based-Multi-agent Pathfinding (WM-MAPF) that incorporates the state representation model trained in Chapter 4 as a centralized communication block which delivers global context to each agent. The detailed training process with components and architecture of the framework is provided, stemming from the theoretical background of Multi-agent Proximal Policy Optimization (MAPPO). The proposed framework demonstrates improved performance in both training process and testing process, highlighting its generalization capability.

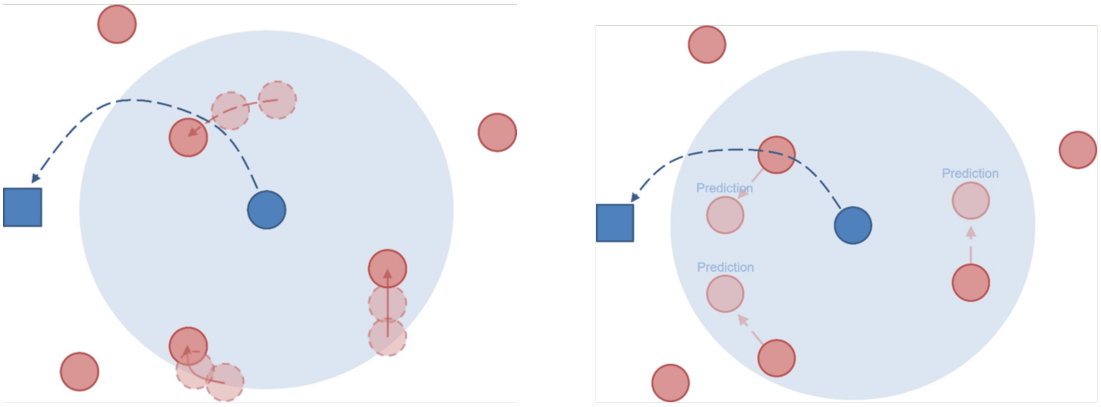
Chapter 6 concludes the thesis with a summary of the main findings of the thesis, discusses promising future researches.

Chapter 2

Background of DRL-based MAPF

2.1 Literature Review

The MAPF problem involves coordinating multiple agents in a shared environment to plan non-conflicting paths from their starting points to their destinations. Originally, "path planning" meant determining the shortest path free of obstacles for an agent to reach its destination. However, in multi-agent environments, optimizing the shortest paths while avoiding all collisions is a NP-hard problem [57]. Nonetheless, extensive research has been conducted to develop optimal and suboptimal MAPF algorithms [19, 26, 85, 86, 119]. Most of these works involve centralized MAPF planners, where the computational demands exponentially increase with the number of agents or the size of the environment, thereby limiting scalability. Although one research has explored rapid planning method using centralized algorithm for large scale environments [67], the authors acknowledge that decentralized approaches may be necessary for real-world applications. Additionally, fully observable assumption in centralized approaches is generally not feasible in industrial settings, where agents typically have limited observations of only their immediate surroundings but still need



(a) Reactive Planner (b) Proactive Planner

Figure 2.1: Diagram depicting two facets of decentralized path-planning approaches.

to coordinate globally.

Distributed approaches offer greater feasibility for applications where agents have partial observability within their sensor range and can effectively distribute the computation, thus enhancing scalability. Consequently, the research community has increasingly studied in enhancing the robustness of decentralized approaches under unstationary observations. Figure 2.1 illustrates that decentralized agents typically make planning decisions on either proactive or reactive methods. In proactive approach, agents' paths are influenced by predictions of future observations. This approach can produce high-quality solutions only if the predictions are accurate; otherwise, unreliable predictions may lead to risky situations. In contrast, reactive approach involves agents adjusting paths based on previous observations. While this approach is generally safer and more computationally efficient, it may result in suboptimal paths or difficulties in navigating complex scenarios effectively.

Recent studies for decentralized planners can mainly be divided into three approaches: search-based methods, bio-inpired meta-heuristics, and learning-based solutions. Among the latter, DRL-based approaches have garnered significant attention due to their ability to address partial observability, enable effective decentralized execution, adapt well to new contexts and uncertainties, and address issues like deadlocks and livelocks. In the following, extensive literature review of aforementioned approaches will be discussed with their advantages.

2.1.1 Search-based Planners

Some approaches partition the complex problem into smaller, more manageable subproblems. They manage the navigation and conflict avoidance of agents while enabling dynamical interactions with the environment [103, 107]. However, they impose constraints or rules on agents' movements, which can compromise the quality

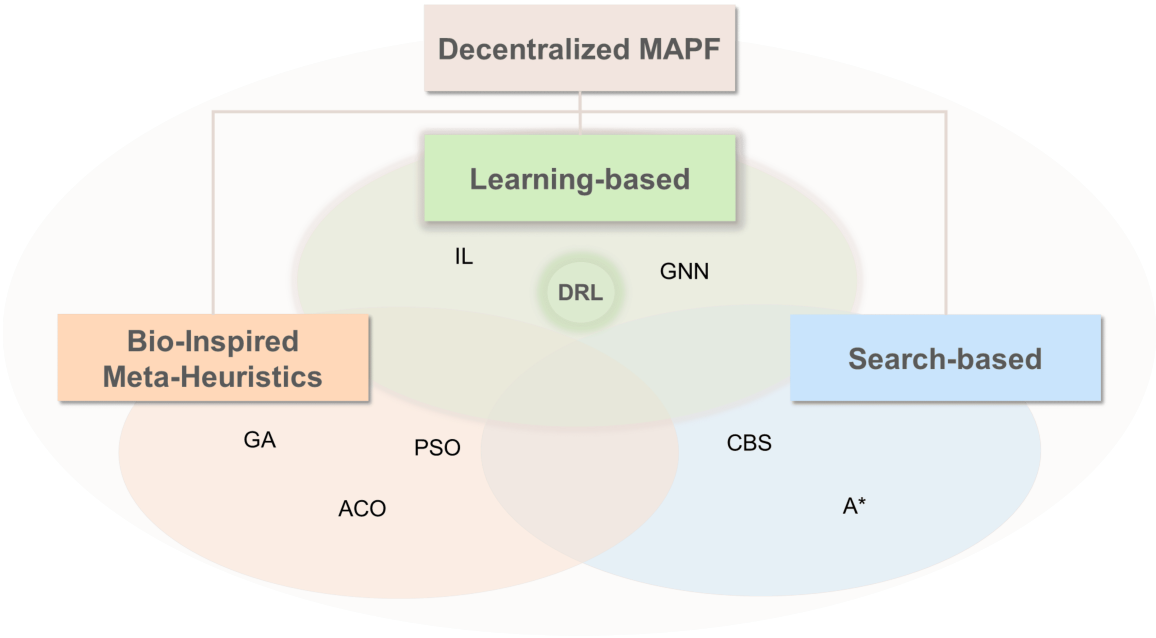


Figure 2.2: Three primary categories of decentralized approaches for MAPF.

of the solutions.

Search-based approaches utilize search algorithms to determine optimal solutions of MAPF. Early approaches in this category were variations of the A* algorithm. [50, 73, 83, 94, 120]. A* algorithm finds the most efficient path from the start node to the end nodes by evaluating the cost of the current node and an estimated heuristic for reaching the goal, thereby prioritizing which paths to explore, making search efficient. Often, A* struggles with large scale MAPF problems due to exponential increase in both the state space and the branching factor as the number of agents grows.

Recent studies on search-based methods for decentralized MAPF have concentrated on adapting a hierarchical mechanism that involves two levels of MAPF. A commonly used technique in this area is the Conflict Based Search (CBS) method [85]. CBS employs a hierarchical search structure: the upper level addresses conflicts between agents, while the lower determines paths for agents based on constraints imposed by the upper level. Originally developed as a centralized solver for MAPF, CBS's adaptability to integrate with other techniques makes it effective for decentralized MAPF applications [48, 66].

2.1.2 Bio-inspired Meta-Heuristics Planners

Bio inspired metaheuristic techniques have become prominent solutions for MAPF as well. Drawing inspiration from natural biological processes, such as Ant Colony Optimization (ACO), Genetic Algorithms (GA), and Particle Swarm Optimization (PSO), these methods tackle optimization problems where achieving optimal or nearly optimized solutions is challenging or impractical. Although these methods do not always ensure a global optima, they may offer good solutions even with partial information, often with reduced computational expense [6].

PSO tackles optimization problems by having a group of potential solutions explore

various areas in the search domatin to discover the best results based on the collective behavior observed in birds or fish. It has recently been adapted for use as a planner in decentralized MAPF [1, 15, 98, 110]. In contrast, ACO, which is based on the foraging patterns of ant colonies, uses probabilistic methods and pheromone signals to determine the most efficient routes to resources. ACO can be also utilized for decentralized solutions [84]. GAs are based on the evolutionary processes that drive biological evolution. They utilize genetic operators, mimicking natural selection and genetic variation, to identify and refine potential solutions for optimal solution. GAs have been employed in decentralized MAPF solutions taking advantage of their high parallelism and the capability finding global point of optimization, often incorporating co evolution mechanisms [65, 71, 100].

While bio inspired metaheuristics are proficient at navigating the search-spaces to improve solution quality, they can become increasingly computationally demanding as the scale of the problem expands [43]. When scaling MAPF to a large environments with numerous robots, addressing the challenge of dimensionality becomes unavoidable.

2.1.3 Learning-based Planners

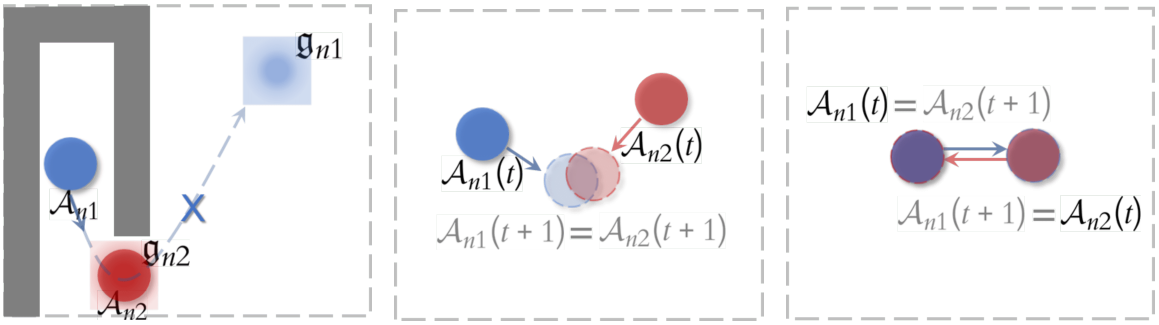
DL and RL have achieved significant progress in developing decentralized planners. Combining both, DRL has quickly gained prominence and demonstrates effectiveness in handling MAPF complexity through the use of network functions for approximation [2, 28].

Effectiveness of using DRL-based planners

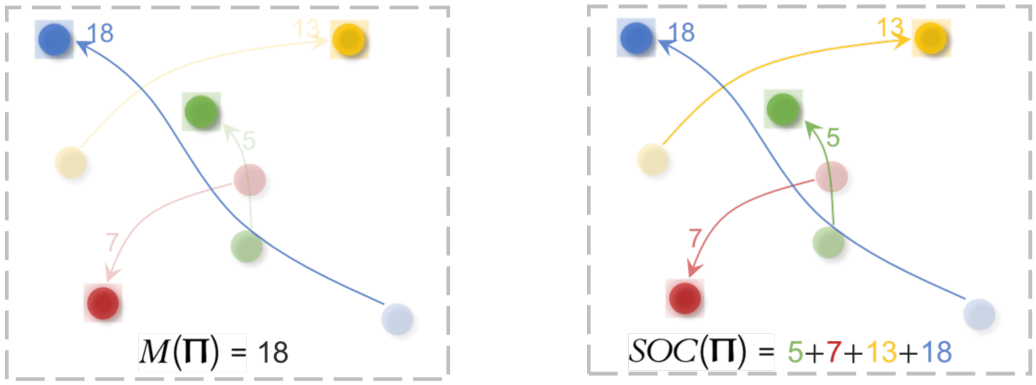
RL is an approach that views the problem objective function as the expected cumulative reward accumulated by an agent interacting with an environment by its actions over a certain time horizon. Therefore, the RL agent learns from its experience how

to optimize its action policy by maximizing the expected cumulative reward. One disadvantage of RL is its difficulty in learning from all states and actions, and storing state action pairs of reward values when handling high-dimensional state and action spaces, which can lead to inefficient learning and poor performance. DRL addresses this issue by using deep neural networks to estimate the objective value functions or policies. These approximation eliminates the need to store all values of state-action pairs in a table, and enables the agents to generalize the value of states from similar states.

In the context of MAPF, the trial and error learning approach improved DRL planners without needing the knowledge of environment's dynamic. Neural network approximators enable agents to formulate decision making strategies using only local observations, thereby reducing the need for computational resources for managing interactions and the reliance on frequent communication. This promotes efficient cooperation and coordination among agents, removing the necessity for explicit coordination mechanisms [9, 51]. Additionally, the flexibility of DRL planners to integrate with other techniques has led to powerful solutions that capitalize on the strengths of each method. For instance, the authors of [81] improved DRL learning by incorporating GA to tune the DRL parameters, leading to more effective learning and performance improvement. Additionally, combining an optimal search algorithm with DRL helps agents reduce the gap between optimal solutions and manage uncertainties [104]. Lastly, DRL-based approaches present a promising solution for tackling the challenges of executing real-time implementations in complex and large-scale environments.



(a) Deadlock Scenario (b) Collision on Vertices (c) Collision on Edges



(d) *Makespan* (e) *Sum of Costs*

Figure 2.3: ((a)-(c)) illustration of failure conditions. ((d)-(e)) two common objective functions for solution optimization.

2.2 Problem Formulation

2.2.1 MAPF

The MAPF problem includes a group of agents, target points and obstacles within a specific map. The main aim is to compute collision free paths that allow all agents to reach their targets from their initial positions. After multiple feasible paths are identified, the problem becomes an optimization problem, focusing on finding the paths that best achieve a specified objective function. From a classical perspective [95], MAPF with N agents, K static obstacles, and M dynamic can be defined as a tuple: $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{O}}_{dyn}, \boldsymbol{\mathfrak{O}}_{static} \rangle$, where $\boldsymbol{\mathcal{A}} = (\mathcal{A}_1, \cdots, \mathcal{A}_N)$ represents the set of N agents, $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ denotes the graph that maps the entire environment, $\boldsymbol{\mathfrak{O}}_{static} = (\mathfrak{O}_1, \cdots, \mathfrak{O}_K)$ represents the set of static obstacles, and $\boldsymbol{\mathcal{O}}_{dyn} = (\mathcal{O}_1, \cdots, \mathcal{O}_M)$ is the set of M dynamic obstacles. Every agent $\forall \mathcal{A}_n \in \boldsymbol{\mathcal{A}}$ is characterized by a tuple $\langle \mathfrak{s}_n, \mathfrak{g}_n \rangle$, where $\mathfrak{s}_n$ and $\mathfrak{g}_n$ represent the start location and goal location of agent $\mathcal{A}_n$, respectively. The vertices $\forall \mathcal{V}_i \in \boldsymbol{\mathcal{V}}$ can be occupied by any agents $\forall \mathcal{A}_n \in \boldsymbol{\mathcal{A}}$ or by any of the obstacles $\forall \mathfrak{O}_k \in \boldsymbol{\mathfrak{O}}_{static}$ and $\forall \mathcal{O}_m \in \boldsymbol{\mathcal{O}}_{dyn}$. Agents and dynamic obstacles can traverse from $\mathcal{V}_i$ to $\mathcal{V}_j, \forall \mathcal{V}_i, \mathcal{V}_j \in \boldsymbol{V}$ via $\mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}}$. Time is discretized into timesteps, allowing each agent to choose a single action to move through an adjacent edge at each timestep. A valid solution to MAPF is defined as a joint sequence of actions $\boldsymbol{\Pi} = (\Pi_1, \cdots, \Pi_N)$, where each $\Pi_n = (a_0^{(n)}, \cdots, a_f^{(n)}) \in \boldsymbol{\Pi}$ satisfies $a_f^{(n)}(\cdots a_1^{(n)}(a_0^{(n)}(\mathfrak{s}_n))) = \mathfrak{g}_n$ without any deadlocks or and collisions. For a solution to be considered valid in MAPF, it should meet the following conditions:

- No deadlock scenario: There exists $T \in \mathbb{Z}^+$ such that $f = T$.
- No collision on vertices: For all $(\Pi_n, \Pi_{n'}) \in \boldsymbol{\Pi}$, and for all $0 \leq t \leq f$,
  $a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)) \neq a_t^{(n')}(\cdots a_0^{(n')}(\mathfrak{s}_{n'}))$,
  $a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)) \notin (\boldsymbol{\mathcal{O}}_{dyn} \cup \boldsymbol{\mathfrak{O}}_{static})$.

- No collision on edges: For all $\Pi_n, \Pi_{n'} \in \mathbf{\Pi}$, and for all $0 \leq t \leq f$,and for all $\mathcal{O}_m \in \boldsymbol{\mathcal{O}}_{dyn}$,
  $(a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)),\ a_t^{(n')}(\cdots a_0^{(n')}(\mathfrak{s}_{n'}))) \neq (a_{t+1}^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_{n'})),\ a_{t+1}^{(n')}(\cdots a_0^{(n')}(\mathfrak{s}_n)))$,
  $(a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)),\ \mathcal{O}_m(t)) \neq (a_{t+1}^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)),\ \mathcal{O}_m(t+1))$,
  where $(a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)),\ a_t^{(n')}(\cdots a_0^{(n')}(\mathfrak{s}_{n'}))),\ (a_t^{(n)}(\cdots a_0^{(n)}(\mathfrak{s}_n)),\ \mathcal{O}_m(t)) \in \boldsymbol{\mathcal{E}}$, and $\mathcal{O}_m(t)$ denotes the vertex occupied by $\mathcal{O}_m(t)$ at timestep t.

objective functions can be used to optimize MAPF solutions when multiple solutions exist. The two common objective functions for MAPF are *Makespan* and *sum of cost*. The *Makespan*, $M(\mathbf{\Pi})$, is the total consumed timesteps for entire agents reachinig goal points:

$$M(\mathbf{\Pi}) = \max_{1 \leq k \leq N} |\Pi_k| \tag{2.1}$$

The *sum of cost*, $SOC(\mathbf{\Pi})$, represents the total comsumed actions required for entire agents reaching goal points:

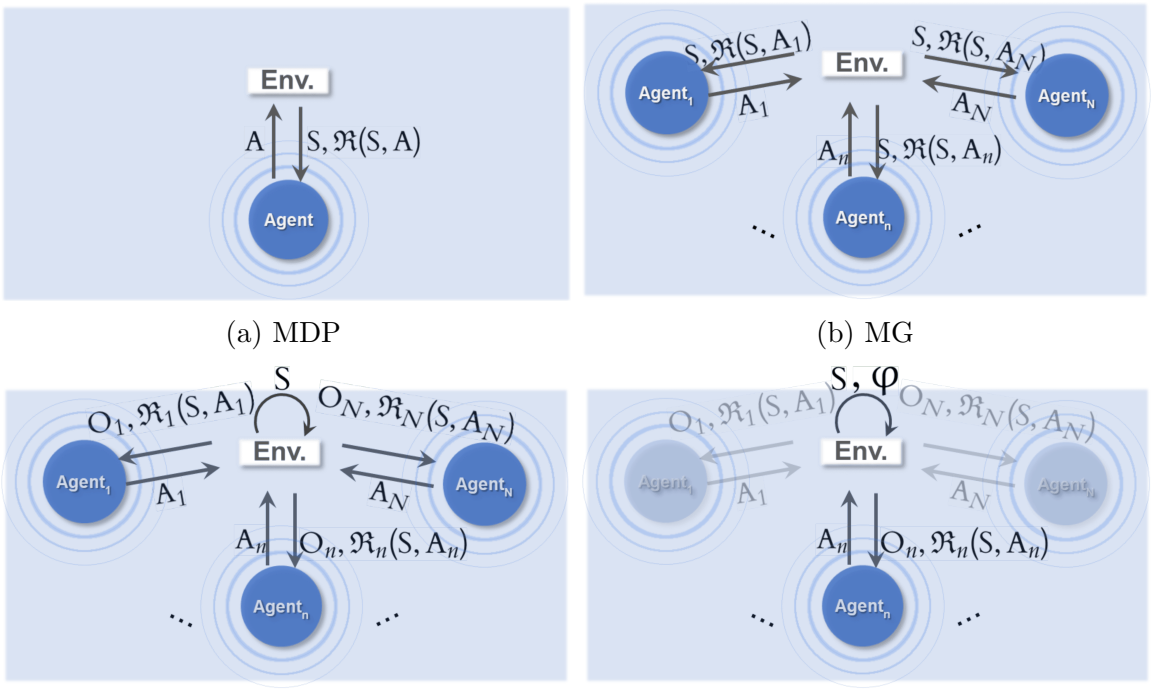
$$SOC(\mathbf{\Pi}) = \sum_{k=1}^{N} |\Pi_k| \tag{2.2}$$

Figure 2.3 illustrated the comditions necessary for a solution to be considered valid and the objective functions used for optimization.

2.2.2 DRL based Framework

DRL based MAPF can be represented as a Markov Game (MG), an extension of the Markov Decision Process (MDP) that incorporates decision making mechanisms for multiple agents in a shared environment [53]. The MDP framework is characterized by a 5 tuple denoted as $< \mathbf{S}, \mathbf{A}, \mathcal{T}, \mathfrak{R}, \gamma >$, where

- $\mathbf{S}$: Represents the set of all possible states.



(a) MDP

(b) MG

(c) POMG

(d) Dec-POMDP

Figure 2.4: Schematic illustration of DRL based MAPF framework.

- $\mathbf{A}$: Represents the set of all possible actions.
- $\mathfrak{T}$: Defines the probability distribution over the next state given the current state and action, mapping : $\mathbf{S} \times \mathbf{A}$ to $P(\mathbf{S},\ \mathbf{A})$.
- $\mathfrak{R}$: Specifies the reward received for taking an action in a given state, mapping $\mathbf{S} \times \mathbf{A}$ to $\mathbb{R}$.
- γ The discount factor, which falls within the range $[0, 1]$.

The objective in an MDP is determining actions that enhance the expected cumulative reward over time, while dealing with an unspecified transition $\mathfrak{T}$ and reward function $\mathfrak{R}$. The agent develops a policy $\pi : \mathbf{S} \rightarrow P(\mathbf{A})$ that seeks to maximize the expected performance G during the learning process. The performance is quantified by the mean cumulative return, computed across the distribution of initial states ρ_0:

$$G = E_{s_0 \sim \rho_0, s_t \sim \mathfrak{T}, a_t \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathfrak{R}(s_t, a_t)\Big]. \tag{2.3}$$

To enhance G starting from a state s, the state-value function $V_\pi : \mathbf{S} \rightarrow \mathbb{R}$ can be used, which evaluates how favorable being in state s with the policy π:

$$V_\pi(s) = E_{s_t \sim \mathfrak{T}, a_t \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathfrak{R}(s_t, a_t) \mid s_0 = s\Big]. \tag{2.4}$$

The agent improves the expected performance G by iteratively refining the policy π maximizing the state-value function. We can also use the action-value function $Q_\pi : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$, or Q-function, to estimate the value of taking a particular action a in state s while following the policy π:

$$Q_\pi(s, a) = E_{s_t \sim \mathfrak{T}, a_{t>0} \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathfrak{R}(s_t, a_t) \mid s_0 = s, a_0 = a\Big]. \tag{2.5}$$

In DRL, neural networks are used to estimate either the policy π or the value functions.

In an N-agent MG, unlike a single-agent MDP, the consideration of multiple agents $\boldsymbol{\mathcal{A}}$ becomes necessary, as elaborated in Chapter 2.2.1. Furthermore, the action space $\mathbf{A}$ evolves into a joint action space $\mathbf{A}_{1...N} = \mathbf{A}_1 \times \cdots \times \mathbf{A}_N$ and the reward function $\mathfrak{R}$ transforms into a collection of reward functions $\boldsymbol{\mathfrak{R}} = \{\mathfrak{R}_1, ..., \mathfrak{R}_N\}$, where each reward function $\mathfrak{R}_n$ is defined as $\mathfrak{R}_n : \mathbf{S} \times \mathbf{A}_n \rightarrow \mathbb{R}$. This expansion requires modification of the state transition distribution function, leading to $\mathcal{T} : \mathbf{S} \times \mathbf{A}_{1...N} \rightarrow P(\mathbf{S})$. Ultimately, a MG is described by the tuple $< \boldsymbol{\mathcal{A}}, \mathbf{S}, \mathbf{A}_{1...N}, \mathcal{T}, \boldsymbol{\mathfrak{R}}, \gamma >$.

Typically, agents do not have direct access to the underlying state required for optimal decision-making, thus requiring a mapping from observational histories or belief states to actions. To tackle this issue, some research has expanded the MG framework to a Partially Observable Markov Game (POMG) [58]. The POMG framework, designed for environments with partial observability, is characterized by the tuple $< \boldsymbol{\mathcal{A}}, \mathbf{O}_{1...N}, \mathbf{S}, \mathbf{A}_{1...N}, \mathcal{T}, \boldsymbol{\mathfrak{R}}, \gamma >$, where

- $\mathbf{O}_{1...N} = \mathbf{O}_1 \times \cdots \times \mathbf{O}_N$ denotes the joint observation space, with $\mathbf{O}_n$ representing the observation space of agent n.
- $\mathcal{T} : \mathbf{S} \times \mathbf{A}_{1...N} \rightarrow P(\mathbf{S} \times \mathbf{O}_{1...N})$ specifies the function for state transitions and observations.

Beyond POMG, various approaches have addressed MAPF using decentralized Partially Observable Markov decision processes (dec-POMDP), providing a more straightforward way to model the problem [22, 29]. In contrast to POMG, which considers the whole environment, dec-POMDP focuses on the individual agent's perspective, incorporating terms for observing other agents and the environment in a decentralized approach. The dec-POMDP framework can be described by the tuple $< \mathbf{O}, \mathbf{S}, \mathbf{A}, \mathcal{T}, \mathfrak{R}, \varphi, \gamma >$, where

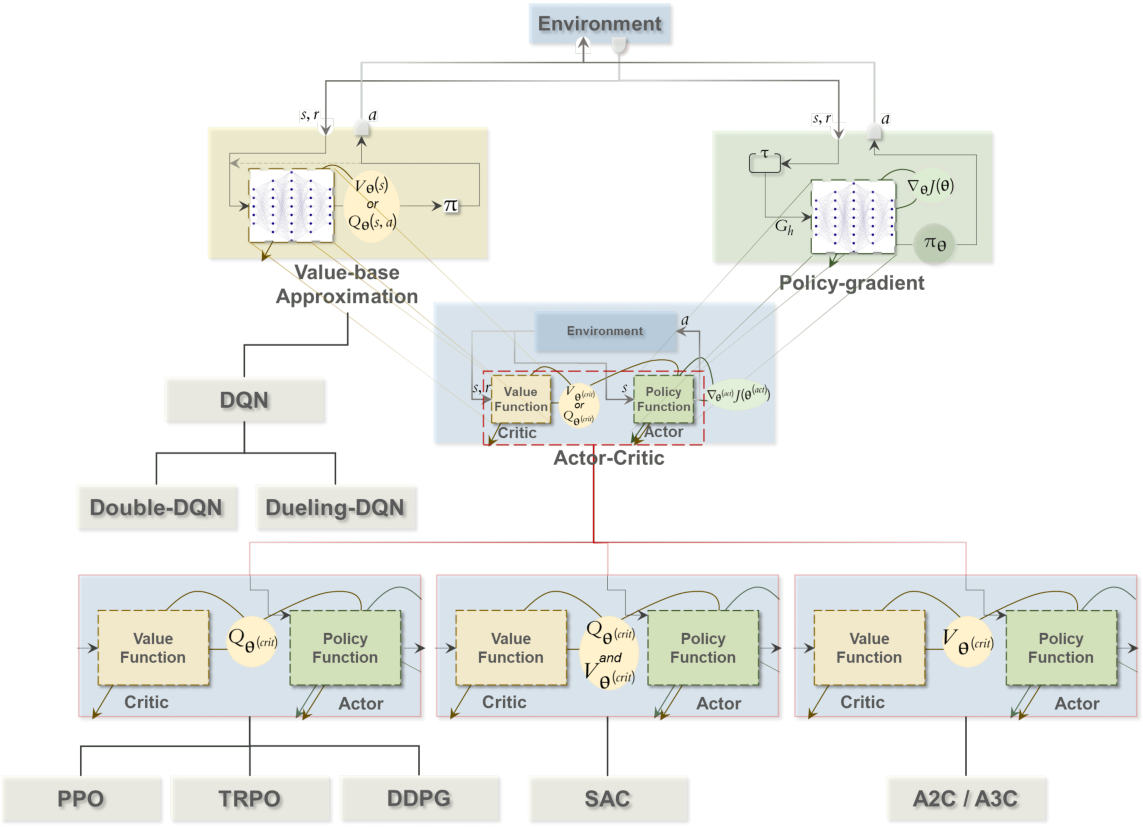


Figure 2.5: A visual summary of the concepts discussed in Chapter 2.3.

- φ represents the conditional observation probability distribution, which includes the additional term.

This approach is beneficial for handling large-scale MAPF challenges, as it reduces the complexity of the MAPF framework. Figure 2.4 presents the frameworks relevant to DRL based MAPF.

2.3 DRL Categorization

Here, we focus on model-free DRL approaches, which rely on interactions with the environment to estimate value functions or define policies directly, without requiring a pre-existing model of the environment. These methods are effective in scenarios

where learning from experience is sufficient, as they avoid the complexities of modeling the environment's dynamics, such as state transition functions $\mathcal{T}$ as denoted in Chapter 2.2.2. These methods utilize networks that process environmental states or observations as inputs, optionally including or excluding the associated actions. Reinforcement learning algorithms train these networks to approximate value functions or directly determine actions. Figure 2.5 illustrates that DRL based approaches are generally divided into two types according to the functions of the networks: value based approximation methods and policy gradient methods.

Value based approximation methods are centered upon discovering the optimal value functions which we discussed in Chapter 2.2.2 (Eq. 2.4, 2.5). These methods aim to progressively adjust network parameters to approach the ideal value function V_{π^*} or Q_{π^*}. A popular value-based approach is the Deep Q-Network (DQN) [61], which leverages deep neural networks to estimate the Q-function Q_π, as shown in Eq. 2.5. In training, experience replay D is used to randomly select a minibatch $E = \{e_1, ..., e_I\}$, where $e_i = (s_i, a_i, r_i, s'_i)$, for i in $\{1, ..., I\}$, represents a (state, action, reward, next state) tuple drawn from the agent's interactions. This method minimizes the correlation among minibatch samples, contributing to more stable training convergence by lessening the effects of changing target distributions. The parameters $\boldsymbol{\theta}$ of the DQN network are adjusted to reduce the loss function, which measures the mean-square error (MSE) between the Q values obtained derived from the experience replay minibatch and the target Q values:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta'}} \sum_{i=1}^{I} \left(r_i + \gamma \max_{a'} Q(s'_i, a'; \tilde{\boldsymbol{\theta}}) - Q(s_i, a_i; \boldsymbol{\theta'})\right)^2, \tag{2.6}$$

Here, γ denotes the discount factor, and $\tilde{\boldsymbol{\theta}}$ refers to the parameters of the target Q-network. These parameters are updated periodically, remaining unchanged between

updates. To improve DQN's performance, several adaptations have been introduced, including DDQN [101], Dueling-DQN [112], and the Rainbow method [40]. DDQN employs two distinct Q-value estimators: One for selecting actions and another for evaluating them, with each network updating the other. This approach helps to mitigate the bias towards maximization and the overestimation problems found in DQN. In contrast, Dueling-DQN uses two distinct streams to estimate both state value and action advantages, while utilizing shared network parameters. This design has demonstrated enhanced accuracy in state value estimation and offers more stable and effective learning compared to DQN. Additionally, the authors of [40] proposed several performance-improving techniques, all of which were incorporated into the Rainbow extension, which is recognized as a leading approach among DQN variants. Although value based methods work well in extensive state spaces, they encounter challenges in handling large or continuous action spaces because of the significant computational demands for policy updates Additionally, the deterministic policy generated by DQN might not be appropriate for scenarios where a stochastic approach is necessary.

Policy-gradient methods, on the other hand, directly represent the policy π using the network, rather than approximating the value function. This is expressed as:

$$\pi_{\boldsymbol{\theta}}(a|\ s) = P(a|s;\boldsymbol{\theta}), \tag{2.7}$$

where $\boldsymbol{\theta}$ denotes the policy parameters and applies to any state s and action a. A prominent policy-gradient algorithm is REINFORCE [114], which employs Monte Carlo Sampling [37]. It estimates the value function through trajectories of the form $\tau = (s_0, a_0, r_1, \cdots, a_H, r_{H+1}, s_{H+1})$, where H represents the number of timesteps from 0 to the end of the episode. This approach allows the network to be updated based on ongoing tasks or segments of an episode. The REINFORCE algorithm optimizes the policy network by improving the value function derived from sampled trajectories.

Instead of relying on value based approaches, the REINFORCE algorithm evaluates the value function by considering the expected cumulative rewards associated with a given trajectory and the policy network. The expected cumulative reward for a given trajectory τ is represented as:

$$G_h = \sum_{t=h+1}^{H+1} \gamma^{t-h-1} r_h, \tag{2.8}$$

and the return is represented as $R(\tau) = (G_0, ..., G_H)$. To optimize the expected return $J(\boldsymbol{\theta})$, gradient ascent is employed, which updates the policy network's parameters. This process refines the network to approach a global or local peak of $J(\boldsymbol{\theta})$, where:

$$J(\boldsymbol{\theta}) = \sum_{\tau} P(\tau; \boldsymbol{\theta}) R(\tau) = E_{\pi_{\boldsymbol{\theta}}}[R(\tau)]. \tag{2.9}$$

The gradient of $J(\boldsymbol{\theta})$ concerning the policy parameters $\boldsymbol{\theta}$ is calculated as follows:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\pi_{\boldsymbol{\theta}}}\Big[R(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau)\Big] = \sum_{h=0}^{H} G_h \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_h|s_h)}{\pi_{\boldsymbol{\theta}}(a_h|s_h)}. \tag{2.10}$$

The policy network parameters $\boldsymbol{\theta}$ at timestep $t+1$ are updated using:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{2.11}$$

Although this approach demonstrates favorable convergence characteristics, it can be hindered by slow learning rates resulting from the high variance in the gradient of R_τ. This problem can be mitigated by incorporating a baseline _into Equation 2.10, such that:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{h=0}^{H} (G_h - b) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_h|s_h)}{\pi_{\boldsymbol{\theta}}(a_h|s_h)}, \tag{2.12}$$

where b is chosen to be independent of the policy parameters to ensure that the

gradient estimate remains unbiased.

This concept leads to the development of actor-critic (AC) methods, which replace the baseline b with a value-based network within the policy gradient framework. AC methods methods combine the advantages of value-based approaches with policy-gradient techniques to mitigate the problem of high gradient variance. They accomplish this by integrating a critic-network, which estimates the value function, and an actor-network, which updates the policy directly. The critic-network helps reduce gradient variance by either approximating the Q function $Q_{\boldsymbol{\theta}}$ or the state value function $V_{\boldsymbol{\theta}}$.

When the critic network estimates the Q function, the parameters of the actor network, denoted as $\boldsymbol{\omega}$ are adjusted based on the evaluation of every potential action within a specific state. This is mathematically represented as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \alpha \sum_{\hat{a}} Q_{\boldsymbol{\theta}}(s, \hat{a}) \nabla_{\boldsymbol{\omega}_t} \pi(a|s; \boldsymbol{\omega}_t). \tag{2.13}$$

This idea has led to the creation of sophisticated algorithms, including Deep Deterministic Policy Gradient (DDPG) ([52]). DDPG learns a deterministic policy directly by using samples from experience replay to address challenges in continuous action spaces.

When the critic network estimates state value function, it helps to compute the advantage function $A_{\boldsymbol{\theta}}$. The advantage function measures how much better or worse a particular action is compared to the expected value of the current state. This is mathematically represented as:

$$A_{\boldsymbol{\theta}}(s, a) = Q_{\boldsymbol{\theta}}(s, a) - V_{\boldsymbol{\theta}}(s). \tag{2.14}$$

In practice, the advantage is often computed using the temporal difference (TD) error δ, which measures the difference between the reward received and the expected future

reward based on the next state:

$$\delta = r + \gamma V_{\boldsymbol{\theta}}(s') - V_{\boldsymbol{\theta}}(s). \tag{2.15}$$

This TD error helps approximate the advantage function in the following:

$$A_{\boldsymbol{\theta}}(s, a) \approx \delta. \tag{2.16}$$

This replaces the term $(G_h - b)$ in Equation 2.12 and is used to improve the policy during training. The update for the actor-network is given by:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \alpha A_{\boldsymbol{\theta}}(s, \hat{a}) \frac{\nabla_{\boldsymbol{\omega}_t} \pi(a|s; \boldsymbol{\omega}_t)}{\pi(a|s; \boldsymbol{\omega}_t)}. \tag{2.17}$$

This method is referred to as Advantage Actor-Critic (A2C). If several agents concurrently interact with different instances of the environment and independently adjust their networks, the approach is called Asynchronous Advantage Actor-Critic (A3C) [60]. A3C allows for broader exploration of the state-action space more efficiently, with each agent performing independent exploration and contributing to updates of the global network. Advanced algorithms such as Trust Region Policy Optimization (TRPO) [78] and PPO [80] share similarities with A2C. TRPO maintains stability and ensures dependable updates to the actor network by applying a trust region constraint to policy adjustments. Conversely, PPO uses a surrogate objective function to limit the extent of policy changes, thereby preventing destabilization.

A recent study discussed in [32] presents a novel application of maximum entropy within the Actor-Critic model. This approach incorporates a soft Q-network, leveraging its Q-values to adjust the state-value estimates. The framework's goal is twofold: to increase the expected rewards and to enhance exploration by maximizing entropy,

thus steering the agent away from less promising actions.

2.4 MADRL for MAPF

DRL methods can approach MAPF by breaking it down into individual path planning tasks for each agent, treating other agents as environmental factors. Although this method streamlines the problem, it may lead to challenges such as failure to manage dynamic changes and complexities, possibly causing collisions or gridlocks. In contrast, a centralized strategy could be used, where DRL manages all agents' actions from a single control point, addressing non-stationarity issues. Yet, this centralized approach can become computationally demanding and impractical for large-scale scenarios.

Multi-agent deep reinforcement learning (MADRL) presents a powerful method that integrates the strengths of various strategies. In the context of MAPF, the actions of multiple agents collectively shape the environment, rather than being influenced by a single agent alone. MADRL utilizes inter-agent communication and collaborative strategies to tackle the intricate nature of these environments. Additionally, MADRL allows for the exchange of knowledge among agents, adhering to the CTDE framework [28]. This method enhances the efficiency of training and speeds up the learning process by enabling the sharing of critical information. [23].

In the realm of MADRL, approaches can be split into two main categories: those focusing on value estimation and those leveraging policy gradients. One notable technique in the value-based category is Value Decomposition Networks (VDN) [96]. This method assumes that the reward obtained by the team can be split into separate Q values for each agent. VDN's structure allows for learning based on collective rewards, while each agent's actions are informed by its individual observations. This method promotes each agent's independent value function development while aiding

the team's overall goal. There are advanced versions of VDN designed to enhance the decomposition of the global Q value, including QTRAN [92, 93], QMIX [72], and QPLEX [106].

QMIX advances the VDN model by integrating a mixing network that processes individual Q values through a non-linear approach. This model assumes that the overall reward must maintain a monotonic relationship between the global and individual Q-values, allowing decentralized agents to effectively use both their local observations and the information from nearby agents to make decisions. On the other hand, QTRAN challenges this monotonicity assumption by redefining the global Q-value function into a form that can be decomposed, leading to better agent coordination than what is achieved with VDN and QMIX. QPLEX further improves upon this by utilizing a duplex dueling network architecture to decompose the global value function. This design enhances the learning efficiency of value functions and demonstrates notable advantages over other value-based approaches, especially in tasks requiring detailed management.

Conversely, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [56] stands out as a prominent policy gradient technique. This method modifies the AC framework by using centralized critics to train the policies and observations of all participating agents, while each agent operates with its decentralized actor that processes individual action-observation pairs. In MADDPG, individual critic networks are used for each agent, although the network parameters are shared. This setup enables agents to obtain varying rewards based on their own observations and actions. Another technique, Counterfactual Multi-Agent (COMA) [23], also builds on the AC model but introduces a counterfactual baseline that assesses the impact of each agent's actions on the global reward, comparing them with alternative actions they could have taken. COMA tackles the credit assignment problem by analyzing

the individual contributions of each agent to the overall reward. A newer method, MAPPO [118] combines a centralized value function with PPO. In MAPPO, both the policy and critic network parameters are shared among agents, and it employs Generalized Advantage Estimation (GAE) [79] as a standard practice for implementing PPO. Research by [118] shows that MAPPO performs exceptionally well relative to other off-policy techniques.

2.5 Recent DRL Approaches for MAPF

In this chapter, we will explore recent developments in DRL-based strategies for MAPF within intricate environments, highlighting significant advancements made over the past five years. There has been a notable surge in research focusing on the evolution and enhancement of model-free DRL techniques for MAPF in recent years. These innovative approaches aim to boost adaptability and efficacy in complex environments where various factors need to be addressed due to their inherent complexity and uncertainty. As outlined in Chapter 2.3, model-free DRL methods are generally classified into two categories: value-based methods and policy-gradient methods. This study will concentrate on recent advancements in both value-based and policy-gradient model-free techniques. Table 2.1 provides an overview of the conditions and performance metrics for model-free methods, categorized by their respective approaches.

2.5.1 Value based Methods

DRL techniques can enhance MAPF solutions by either incorporating prior knowledge into the network or optimizing its architecture. For example, a recent study proposed a method to address the issues of slow convergence and excessive randomness in action

Table 2.1: Summary of recent DRL methods, their specific conditions, and performance metrics.

DRL Algorithm		Paper	Decentralized Execution	Real-Robot Application	Existence of Dynamic Obstacles	Observability	Environment Space	Maximum Number of Agents Tested	Success Rate under Maximum Agents	Detour Percentage	Degree of Freedom	Crowd edness	Github link	Video link
Value-Based	DQN	[58]	✓			Part.	Grid	64	73	-	0.185	0.0143	link	-
		[59]	✓			Part.	Grid	128	82	-	0.185	0.0286	link	-
		[10]	✓			Part.	Grid	64	85	-	0.185	0.0571	-	-
	DDQN	[104]	✓		✓	Part.	Grid	128	99.7	9.60	0.185	0.08	-	link
		[117]				Full.	Grid	8	-	-	0.185	0.0148	-	-
	Dueling-DQN	[105]				Full.	Cont.	4	92.3	-	0.333	-	-	-
Actor-Critic	PPO	[55]	✓		✓	Part.	Cont.	20	96.5	9.50	0.333	-	-	link
		[22]	✓	✓	✓	Part.	Cont.	20	100	5.80	0.333	0.0354	-	link
		[113]	✓		✓	part.	Cont.	4	96	-	0.333	-	-	-
		[39]	✓		✓	Full.	Cont.	7	100	-	0.333	-	-	-
	DDPG	[8]	✓			Part.	Cont.	4	92	0.333	-	-	link	
		[70]				Full.	Cont.	5	-	-	0.333	-	-	-
		[30]	✓			Part.	Cont.	4	-	-	0.333	-	-	-
		[41]				Full.	Node	5	-	-	0.185	0.037	-	-
	AC	[29]	✓		✓	Part.	Grid	150	99.1	-	0.333	-	-	-
	A2C	[54]	✓		✓	Part.	Grid	150	94	-	0.333	-	-	-
	A3C	[77]	✓	✓		Part.	Grid	1024	96	-	0.185	0.04	link	link
		[14]	✓	✓		Part.	Grid	2048	-	-	0.185	-	link	link
		[51]	✓			Part.	Grid	64	-	-	0.185	0.08	-	-
		[7]	✓			Part.	Grid	1024	100	-	0.148	0.04	link	-
		[82]	✓		✓	Part.	Cont.	16	100	-	0.333	-	-	-

The table includes the following abbreviations: Part. for partially observable; Full. for fully observable map space; and Cont. for continuous map space.

selection associated with DQN [117]. The approach involved initializing the Q-network with insights gained from a single automated guided vehicle in a static setting, using the A algorithm. This strategy provided the agents with a foundational understanding of their environment, leading to more efficient learning. Additionally, predefined rules were introduced to direct agent behavior and limit unnecessary exploration within the DQN framework. Another advancement highlighted by [105] involves the Duel neural network (Duel-NN), which has been shown to surpass both DQN and DDQN in MAPF scenarios. Unlike DQN and DDQN, Duel-NN features an advantage function stream alongside the value function stream, helping to mitigate overestimation issues by reducing gradient variance and bias. Despite these advancements, challenges such as long-term planning remain prevalent in model free approaches.

To tackle the challenge of long-term planning in MAPF, designing a new reward system that promotes favorable behaviors in scenarios prone to collisions or deadlocks can be effective. For instance, [104] introduced a globally guided reinforcement learning approach featuring an innovative reward structure. This structure utilizes spatio-temporal data from diverse environments to enhance generalization. The proposed reward function aligns with the global path, enabling agents to follow a more efficient route while effectively managing potential collision scenarios. Importantly, this method was implemented in a fully decentralized manner without communication between agents, achieving reliable results across various environments, regardless of complexity or uncertainty. The performance levels were comparable to those of advanced centralized methods. However, as environments become more intricate, solely depending on manually crafted reward structures may not always ensure optimal decision-making by the agents, especially in novel or unpredictable situations.

Another effective strategy for addressing long-term planning issues is to enable communication among agents. This approach can mitigate non-stationarity arising

from incomplete observations and reduce the likelihood of collisions or deadlocks. For example, [58] enhanced the duel-DQN framework by integrating Multi-Head Attention mechanisms [102] and Gated Recurrent Units [12] into the communication component. This advancement facilitated more effective inter-agent communication and collaboration through graph convolution techniques. In contrast to earlier methods that relied on single-agent shortest path guidance [29, 54, 104], their approach, termed DHC, incorporated heuristic guidance through a four-channel field of view, thereby broadening the selection space based on shortest path embeddings. DHC proved effective in large-scale, long-range environments. Nonetheless, combining broadcast communication with graph convolution demanded substantial bandwidth and led to redundant data transmission. To address this, [59] introduced a 'request-reply' communication model within the DHC framework, enabling agents to focus on exchanging only pertinent information for decision-making. Their updated framework, named DCC, showed enhanced performance by better identifying and leveraging relevant neighbors for cooperation in MAPF.

In contrast, the approach from [10] incorporated a self-attention mechanism into their policy network to capture implicit collaborative signals among agents based on their observations. This distinguishes their method from those in [58, 59], as their agents operate reactively without direct communication. To promote coordinated learning among agents, they introduced a novel hot supervision contrastive loss derived from an expert planner, integrating it with the standard RL loss function for the policy network. This combined loss function supported the agents in developing effective coordination strategies.

2.5.2 Policy Gradient Methods

By leveraging initial knowledge of environmental dynamics, the authors from [30] enhanced the decision-making efficiency and accelerated the convergence of DRL agents by integrating artificial potential fields (APF) with the DDPG algorithm. Similarly, for mapless navigation, [42] combined DDPG with Prioritized Experience Replay (PER), which resulted in quicker convergence, decreased fluctuations, and the creation of safer, smoother trajectories. Nevertheless, these methods fall short in addressing challenges related to partial observability, especially in complex, long-range planning scenarios.

Despite its widespread application in multi-agent scenarios, where the framework of CTDE is utilized, DDPG has shown notable adaptability—known as MADDPG. In MADDPG, a centralized critic addresses non-stationary conditions, while decentralized actors follow policies based on their private observations. In the context of scheduling automated guided vehicles (AGVs) at container terminals, [41] integrated MADDPG with the Gumbel-Softmax technique. Meanwhile, [8] employed MADDPG in a Delay-Aware Markov Game framework to mitigate performance issues related to delays and non-stationarity inherent in multi-agent systems. For mapless navigation [42] combined DDPG with Prioritized Experience Replay (PER) to achieve faster convergence, minimized fluctuations, and smoother, safer trajectories. Additionally, [70]applied MADDPG to develop an algorithm for simultaneous target assignment and path planning, incorporating rewards for both tasks to optimize distance and avoid collisions. This centralized training with decentralized execution approach is also adaptable to other algorithms; for instance, [39] employed asynchronous multithreading in PPO (AMPPO) for underwater unmanned vehicle applications.

Another well researched approach in policy gradient methods involves PPO-based algorithms. The researchers in [55] advanced MAPF to a multi scenario, end to end

sensor level decentralized MAPF in continuous spaces. They utilized a PPO to translate raw LiDAR sensor inputs into steering commands, operating under a framework of centralized learning with decentralized execution. To improve the training process, they adopted the curriculum learning paradigm, as discussed in [5], which facilitated more efficient convergence to effective solutions. Although their research showed success in navigating multi-robot systems in extensive simulated environments, it fell short when applied to actual mobile robots. To bridge the gap between simulation and real world application, they developed a hybrid control system that integrated their existing RL model with traditional control techniques [22]. This progression significantly advanced the validation of DRL based algorithms for MAPF, which were mainly tested in simulation. In a similar vein, the authors from [113] incorporated covariance matrix adaptation evolutionary strategies into the PPO algorithm to refine policy parameter updates. Additionally, they included meta reinforcement learning and transfer learning strategies within the A3C framework, which improved robot navigation in unfamiliar settings, resulting in enhanced learning efficiency and shorter training times.

Despite the progress in DRL based MAPF methods, challenges continue, especially with long range planning in extensive, crowded environments. One strategy to manage this involves breaking the problem into smaller, more manageable tasks. In [54], the researchers used global waypoint coordinates as inputs for an A2C framework. By applying an evolutionary algorithm [89] to update the networks, they achieved stable convergence, even in large scale settings. Their study showed significant improvements in DRL-based MAPF for numerous agents, particularly in handling dynamic obstacles. Furthering this work, the researchers in [29] incorporated a BicNet [68] into the actor network design and added an attention mechanism to the critic-network, enhancing stability and performance. Their algorithm outperformed the MAPPER algorithm in

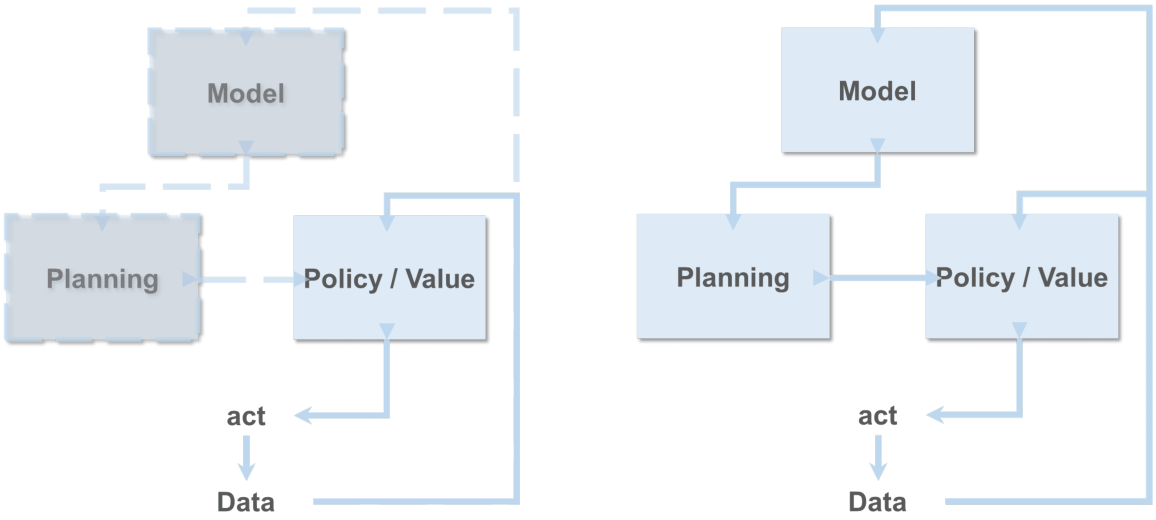
comparative tests. These methods indicate promising ways to overcome the challenges of long range planning in large scale MAPF scenarios.

Deadlocks represent a significant problem for numerous agents, especially in compact and complex environments like mazes. The introduction of rotational movement can contribute to deadlocks, as it expands the agents' search space from two to three dimensions. To tackle this deadlock scenario, the researchers from [7] created a deadlock-breaking method that integrates an imitation learning (IL) expert with A3C, incorporating well-designed input maps and rewards. Their experiments showed that this method effectively resolves MAPF issues involving rotational movements. A further challenge that can cause deadlocks is a fully decentralized framework, which makes it challenging to promote selfless actions among agents [87]. The research from [51] proposed a solution by combining decentralized path planning with centralized collision avoidance. They introduced a prioritized communication learning method (PICO), which merges imitatively learned implicit planning priorities with a communication learning strategy. PICO has shown its effectiveness in improving success rates and reducing collision rates, especially in large-scale scenarios.

DRL-based MAPF algorithms often encounter issues with agents displaying selfish behaviors. To mitigate this, researchers have developed hybrid frameworks that integrate RL with other methodologies to address critical scenarios where DRL policies may not yield optimal decisions. For example, the researchers from [82] proposed a hybrid framework that alternates between force-based motion planning and GPU/CPU A3C for Collision Avoidance with DRL [20], depending on the situations faced by the robots. The researchers from [91] introduced a hybrid policy that combines two strategies, one from heuristic search and the other from RL, using a switching mechanism. The work from [77] proposed a hybrid training framework that includes a 'Blocking Penalty' to prevent agents from pursuing their goals while blocking others.

They also incorporated an IL expert into the RL framework, merging off-policy behavior cloning with on-policy methods. To ensure agents encountered a variety of scenarios, they randomized the map's size and obstacle density during training. This algorithm was successfully implemented on physical robots in a factory mockup, demonstrating that their agents could learn cooperative behaviors. Furthermore, the authors from [14] enhanced PRIMAL to perform well in highly structured and constrained environments with lifelong tasks, using conventions to improve implicit agent coordination.

The researchers from [111] advanced the previous work (PRIMAL) by integrating a transformer-based communication learning mechanism into the PRIMAL framework. This addition effectively resolves the chatter problem caused by conflicting messages from a large number of agents. This communication method not only supports global information exchange among agents, promoting cooperation, but also enables agents to learn from the historical observations of their peers, addressing the partial observability issue common in decentralized systems. Their findings indicate that this communication mechanism provides agents with ample information, even when their observation range is limited. Moreover, the researchers from [38] utilized a graph transformer architecture, allowing agents to access approximate global information and make short-term predictions about other agents' intentions. Their ablation study revealed that this approach significantly improves coordination among agents and reduces episode duration.



(a) Model-free method (b) Model-based method

Figure 2.6: A comparative flowchart of model free and model based methods.

2.6 A Potential Research Frontier: Model based Approach

It is important to recognize that there are several gaps in MAPF that need to be addressed. Some of the key challenges include managing heterogeneous agents, applying Sim to Real techniques, developing effective communication and coordination skills, tackling credit-assignment issues, and ensuring robustness and safety. These topics are currently being explored as future research directions [28, 115, 116, 121, 123]. In this Chapter, we have opted to focus on the model based DRL for MAPF, a promising area that has not been as extensively studied as other future directions. Although there is increasing interest in the MAPF problem, there have been very few attempts to utilize model based DRL methods to tackle it. The relative newness of the model-based approach, along with the challenges in obtaining accurate models of the environments, has hindered the widespread adoption of such methods in MAPF.

In a recent study, the researchers in [35] introduced a versatile and scalable model

based technique that effectively learns to accomplish long term objectives in Minecraft autonomously, without human guidance or predefined heuristics. This research has generated interest in the application of model based RL methods across various fields [18, 99, 108]. Similarly, this model-based approach holds promise for addressing the complexity of large-scale and long-horizon MAPF scenarios, enhancing planning capabilities and improving learning processes. By combining DRL with explicit or learned environment models, model-based DRL enables agents to simulate and predict the consequences of their actions like proactive planners and hence facilitates informed decision-making and coordination.

Figure 2.6 illustrates the distinction between model free and model based frameworks. Model free methods depend on historical and current observations or states, utilizing single step interactions that often label them as reactive planners for MAPF ([10, 17, 21, 104]). Conversely, model based approaches involve an initial planning phase with an environment-aware model, enabling agents to make decisions in a broader context [63]. We view the model-based MAPF solver as a proactive planner, which combines i) a future planning phase [76] with a learned model and ii) a policy update phase through DRL to estimate the global value or policy function in MAPF.

A model-based approach can significantly reduce the vast number of samples needed for effective learning in MAPF. This method offers several advantages, including decreased exploration time, improved sample efficiency, a better grasp of complex dynamics, and enhanced adaptability to dynamic environments [109]. Consequently, it is expected to improve coordination and planning in MAPF. The next Subchapters will explore two essential steps for model-based methods: learning the dynamics model and integrating planning with learning.

2.6.1 Learning Dynamics Model

The initial crucial step in the model based approach for MAPF is learning the dynamics model. This involves addressing a supervised learning task to determine the state transition probabilities and associated rewards. In MAPF, with a batch of one-step transitions represented as $\{s_t, a_t, r_t, s_{t+1}\}$ where s_t is the state, a_t is the action, r_t is the reward, and s_{t+1} is the next state, the dynamics function can be modeled in various ways [64]:

- ***Standard Markovian transition model :*** $(s_t, a_t) \rightarrow (s_{t+1}, r_t)$. Typically, the observation directly represents the state.

- ***Partial observability model :*** $((s_{t-l+1}, ..., s_t), a_t) \rightarrow (s_{t+1}, r_t)$. Here, the current observation's lack of information is compensated by using the history of previous observations.

- ***Multi step prediction model :*** $(s_t, (a_t, ..., a_{t+l-1})) \rightarrow (s_{t+l}, \sum_t^{t+l-1} r_t)$. Using multi step predictions by repeatedly applying single step models can introduce errors since the single step model isn't optimized for long term predictions, leading to potential inaccuracies in multi step forecasts.

- ***State abstraction model :*** $(z_t, a_t) \rightarrow (z_{t+1}, r_t)$. This model maps observations to a compact latent representation, similar to an autoencoder [74], reducing computational complexity and simplifying planning and policy updates.

- ***Temporal/action abstration model :*** $(s_t, u_t) \rightarrow (s_{t+l}, \sum_t^{t+l-1} r_t)$. This approach defines a high level action space spanning multiple timesteps, akin to the multi step prediction model. It is frequently used in hierarchical RL, where the model learns an abstract action u_t.

Neural networks, capable of managing high dimensional inputs and accurately approximating non linear functions throughout the state space, are commonly used to learn these models.

The World Model introduced by [31] has recently gained significant attention due to its expressive capabilities and its proficiency in learning complex spatial and temporal data representations. This model enables the agent to understand environmental dynamics by reducing high dimensional data to a lower dimensional format, focusing only on key environmental features without human guidance. In MAPF, the World Model can abstract global information by modeling the comprehensive context of agents from partial observations. For example, modeling communication blocks as discussed in [111], or predicting agents' intentions as detailed in [38], can facilitate more extensive scenarios. We believe that using this approach could significantly improve agents' abilities to coordinate intelligently and grasp spatial contexts in MAPF.

2.6.2 Planning and Learning Integration

The subsequent crucial step involves integrating the planning and policy updating stages. The planning phase encompasses decisions on when and how to start planning, how long to plan, and the specifics of the planning process. For instance, in the Dyna framework [97], planning occurs only at previously encountered states, where the model generates 100 one-step transitions for each action in every iteration. Conversely, AlphaGo Zero [88], directs its planning efforts at the current state, performing 1600 iterations of Monte Carlo Tree Search (MCTS) [13] at a depth of 200.

Specifying the planning method is also crucial at this stage. Traditionally, the AI and RL fields have favored discrete planning techniques, including probability limited search [49], breadth limited and depth limited search [25], and MCTS [88].Alternatively,

differential planning methods can be used, taking advantage of the learned model's differentiability for gradient based optimization. Notable examples of differential planning approaches are iterative linear quadratic regulator planning [4], PILCO [16], and DreamerV3 ([33, 34, 35]). This paper outlines a general framework for these planning methods, and for an in-depth examination of recent advances, readers are directed to [62].

After setting up the planning stage, the following task is to link it with the policy updating phase of the learning process. The results from the planning phase can be used to adjust the value or policy function in a model free DRL network. In the predictive world model covered earlier, the value or policy function operates as a control model, using a compact and straightforward neural network to extract pertinent environmental information and steer the decision making process.

Chapter 3

Experimental Setup and Data Collection

3.1 Vectorized Multi-Agent Simulator as a environmental setup of MAPF

The Vectorized Multi-Agent Simulator (VMAS) is an advanced open-source framework specifically aimed at providing efficient benchmarking platform in MARL. It leverages a vectorized 2D physics engine implemented in PyTorch, which facilitates the execution of numberous parallel environments on accelerated hardware. This vectorization facilitates the execution of $30,000$ parallel simulations in under 10 seconds, achieving speeds over 100 times faster than OpenAI's MPE. This exceptional efficiency, combined with its straightforward design, establishes VMAS as a potent tool for MAPF research and applications.

VMAS provides twelve challenging multi-robot scenarios that evaluate various dimensions of collective learning, including behavioral heterogeneity, coordination through communication, and adversarial interaction. The platform supports a range

of customizable sensors, such as LIDARs, and allows for the straightforward implementation of additional scenarios through a modular interface. Furthermore, VMAS is compatible with OpenAI Gym and RLlib, enabling the integration of a wide array of reinforcement learning algorithms. This blend of performance, flexibility, and usability positions VMAS as a significant advancement in the field of multi-agent simulation and learning. The key characterists of VMAS can be summarized as:

- **Vectorized:** VMAS vectorization is capable of executing 30000 parallel simulations in under 10 seconds. This enables 100 times faster than OpenAI's MPE.
- **Flexibility:** VMAS supports diverse agent and landmark shapes, torque, elastic collisions, and customizable gravity.
- **Customizable Scenarios:** VMAS includes twelve challenging scenarios with the capability to create new multi-agent scenarios within a modular interface. It also allows debugging scenarios with interactive rendering of the scenarios.
- **Inter-agent Communication:** VMAS supports customizable sensors, including LIDARS, for enhanced inter-agent interaction.
- **Compatibility:** VMAS has multiple wrappers which make it directly compatible with different RL interfaces, such as RLlib, TorchRL, and Gym.

The *navigation scenario* within VMAS specifically targets MAPF challenges, serving as a realistic platform for data collection and testing the proposed MAPF algorithm in this thesis. It emphasizes realistic constraints such as partial observations through agents' LIDAR readings, and the implementation of lower-level 2D applied forces as the action space. In contrast, MPE introduces broader constraints where agents possess object classification abilities to distinguish between obstacles and other agents. Furthermore, the actions of MPE agents directly determine the movements of agents' next positions without considering the physical dynamics like velocities or

accelerations. This distinction makes VMAS an ideal testbed for evaluating navigation solutions across various applications.

The Figure 3.1 and Table 3.1 shows the constraints and conditions of VMAS used in this thesis. Agents are randomly spawn within a 2D square map ranging from -1 to 1. The environment is obstacle-free and can render the entire map with a 64×64 pixel images, assuming a camera vertically projecting the whole area. Each agent has a radius of 0.02 and utilizes a LIDAR sensor with a range of 0.1 and a resolution of 12 dimensions. The observations for each agent include the displacement from its goal in the x and y dimensions, along with the 12-dimensional LIDAR readings. The action space for agents is continuous, with actions represented as 2D forces $(F_x,\ F_y)$ within the range [-1, 1]. Elastic collisions are enabled between agents, influencing their movements dynamically. The reward structure encompasses positional rewards, collision penalties, individual goal rewards, and team goal rewards. Positional rewards are based on how effectively agents move towards their goals compared to their previous positions. Collision penalties are applied when agents collide with each other. Individual goal rewards are granted when an agent successfully reaches its goal, while team goal rewards are earned when all agents in the scenario achieve their respective goals. These designed conditions and constraints provide a framework for testing and evaluating MAPF algorithms.

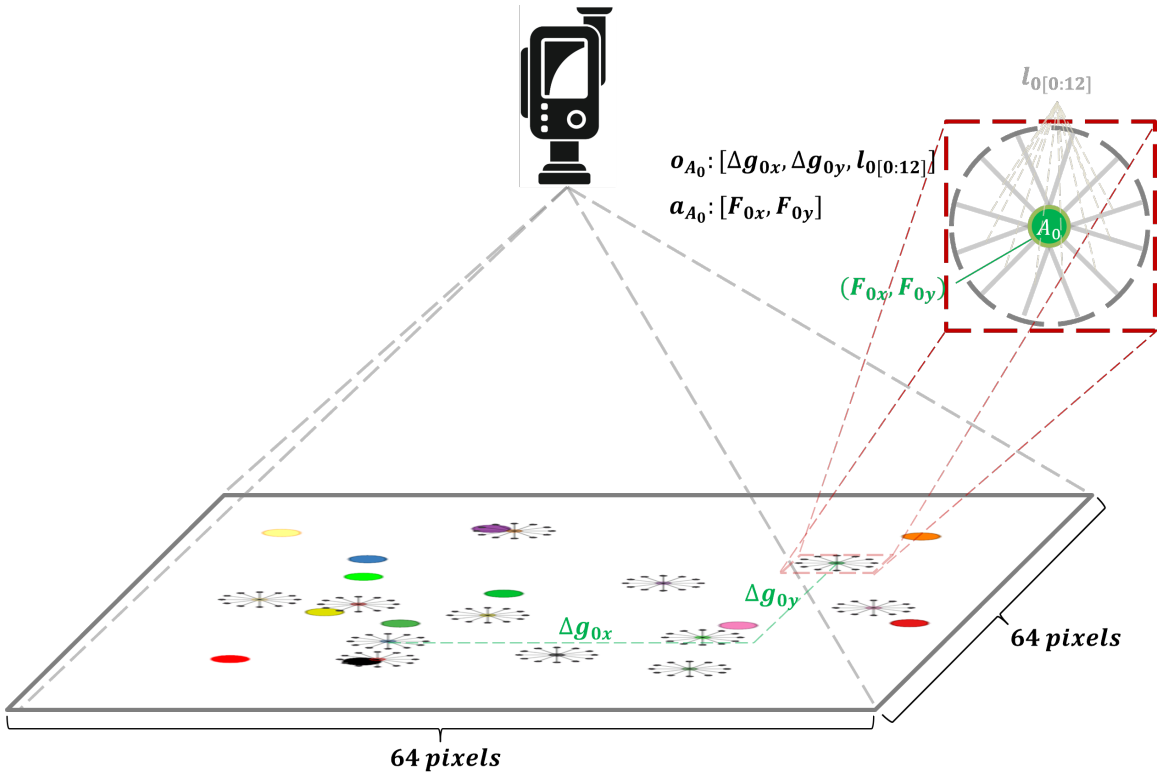


Figure 3.1: Configuration of VMAS setup. The entire map image is rendered in 64×64 RGD image every timestep. A_0 denotes agent 0, Δg_{0x}, and Δg_{0y} denotes the displacement of A_0 from its goal, F_{0x} and F_{0y} denotes the force applied to A_0 in x axis and y axis, $l_{0[0:12]}$ denotes LIDAR sensor readings of A_0, o_{A_0} denotes the partial observation of A_0, and a_{A_0} denotes the action of A_0.

3.2 Data Collection for Training a State Representation Model

The objective of the state representation model is to extract a global context from rendered global map images. To collect enough data for the state representation model to sufficiently learn the global context of a variety of states, VMAS *navigation scenario* was executed with a varying number of agents. The number of agents ranges from 2 to 64. For each scenario, 128 vectorized episodes were simulated, resulting in a total of 63×128 episodes. During each episode, agents performed random actions to explore a wide range of states, with each episode length composing 128 timesteps.

Numbering Style	**Output**
agent location space	$\{x \in [-1,\ 1],\ y \in [-1,\ 1]\}$
agent radius	0.02
LIDAR sensor range	0.1
action space	$\{F_x \in [-1,\ 1],\ F_y \in [-1,\ 1]\}$
Reward Structure	
positional reward	$\Delta g(t-1) - \Delta g(t)$
collision penalty	-1
individual goal reward	10
team goal reward	10

Table 3.1: Conditions of VMAS and the reward structure.

Conducting random policy over agents enables exploring states which would help the state representation model generalizing the global context and extracting essential context features from global map image.

At each timestep, the global map image data was recorded, leading to a dataset comprising $63 \times 128 \times 128$ scene. The dataset was then divided into training and validation datasets, ensuring both comprise different episodes. The training dataset was employed to update the network parameters of the state representation model, while the validation dataset was used to assess the model's performance independently of the training process, thereby mitigating the risk of overfitting.

The dataset was split into training and validation datasets in a 90:10 ratio, leaving training dataset with 7258 episodes and validation dataset with 806 episodes. Prior to splitting, the entire dataset was randomly shuffled to ensure an even distribution of episodes with different numbers of agents across the validation dataset. The data preparation process ensures the robustness and generalizability of the state representation model, facilitating its ability to extract meaningful global context from diverse scenes.

Chapter 4

World-Model as State Representation Model

DRL has demonstrated significant potential in tackling complex sequential decision-making problems. However, it often faces challenges in long-term planning and dynamic modeling problems, particulary when dealing with a vast state space. The policies of DRL planners are determined by their interactions with the environment, which can lead to undesirable actions when encountering perviously unseen states. Therefore, it is crucial to expose DRL planners to a diverse set of states during training to properly refine their policies. This requirement extends the training duration as the state space grows.

One promising approach to mitigate this issue is to abstract the state space by eliminating redundant information, thereby making the state space more concise. This chapter provides a detailed explanation of the proposed state representation model designed to achieve this abstraction effectively.

4.1 Introduction

In recent years, supervised neural networks have been developed and successfully employed to produce representations that have significantly advanced classification accuracy across various tasks. These networks often utilize convolutional architectures in confunction with novel regularization techniques from the deep learning community, such as dropout or maxout.

However, an important question remains: can equally "powerful" representations be learned from unlabeled data without supervision? Despite the strides made in supervised representation learning, it is widely recognized that unsupervised learning algorithms capable of extracting useful features are essential for solving problems with limited or weak label information, such as video recognition or pedestrian detection. One notable advantage of unsupervised learning algorithms is their applicability in semi-supervised scenarios, where the availability of labeled data is restricted.

An autoencoder is a type of neural network that aims to learn efficient representations by reconstructing its original input. It consists of two main components: the encoder and the decoder. Initially, the input vector $\boldsymbol{x}$ is transformed into a hidden representation $\boldsymbol{z} = f(\boldsymbol{x})$ through the encoder network. The function f is parametrized by one or more layers of non-linearity, which allows the network to capture complex patterns in the data. This hidden representation $\boldsymbol{z}$ serves as a compressed version of the input, ideally retaining the most salient features.

Following this, the hidden representation $\boldsymbol{z}$ is mapped to the output $\hat{\boldsymbol{x}} = g(\boldsymbol{z})$ using the decoder network. The decoder function g is also parameterized by multiple layers of non-linearity, mirroring the structure of the encoder. The purpose of the decoder is to reconstruct the input data from the hidden representation as accurately as possible. The parameters of both the encoder and decoder networks are optimized

to minimize the reconstruction error, which is quantified by the following cost function:

$$\mathcal{L}(\boldsymbol{x},\ g(f(\boldsymbol{x}))) = ||\hat{\boldsymbol{x}} - \boldsymbol{x}||_2^2 \tag{4.1}$$

This cost function measures the mean squared error between the input $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$, encouraging the network to produce outputs that are as close as possible to the original inputs. To prevent the autoencoder from learning a trivial identity function that simply maps the input to the output without capturing meaningful patterns, additional constraints are often imposed on the cost function. The constraints can limit the representational capacity of the autoencoder, ensuring that it learns useful and generalizable features. One common approach is to constrain the architecture by limiting the dimensionality of the latent space $\boldsymbol{z}$. Another method is to include regularization terms in the final objective, such as a sparsity constraint, which encourages the hidden representation to use only a few active neurons.

Through this process, the autoencoder learns to capture the essential features of the data in its hidden representations, which can be useful for various tasks such as dimensionality reduction, anomaly detection, and as a pre-training step for other machine learning models. By balancing the reconstruction accuracy and the constraints on the model, autoencoders are able to extract meaningful and compact representations.

World Model

World models draw inspiration from cognitive science, where it is hypothesized that humans and animals construct internal models of the world to facilitate decision-making and problem-solving. These internal models enable organisms to predict the consequences of their actions, learn from past experiences, and adapt to new situations

effectively. World models aim to replicate this ability by employing advanced machine learning techniques to build and refine internal representations of the environment.

World model involves various machine learning approaches, including deep learning and reinforcement learning. The development of world models has been particularly influential in advancing the capabilities of reinforcement learning agents, allowing them to learn and adapt more efficiently by simulating interactions with the environment.

Challenge of Autoencoders

One approach to leaning useful representations with autoencoders is to limit the capacity of the autoencoder by constraining its code size. By doing so, the autoencoder is compelled to extract the most salient feature from the data. It can be demonstrated that if an autoencoder uses linear activation along with the mean squared error criterion, the resulting architecture is equivalent to the principal component analysis algorithm (i.e., the hidden units of the autoencoder learn the principal components of the data). However, autoencoders that employ non-linear activations possess a much larger capacity and can therefore learn more nuanced representations.

Still, deeper autoencoder architecture can face significant challenges, particularly the gradient vanishing problem. This issue is prevalent in the training of deep neural networks, especially those with many layers or recurrent structures. The gradient vanishing problem arises when the gradients of the loss function with respect to the model parameters become exceedingly small as they are propagated backward through the network during training. Consequently, the updates to the model parameters become negligible, effectively halting the learning process for the earlier layers of the network. As a result, these layers receive minimal updates, while only the latter layers close to the output experience learning. This issue is exacerbated by activation functions such as the sigmoid or hyperbolic tangent, which can squash

input values into a small range, thereby further diminishing the gradient. To address this problem, various techniques have been proposed, including advanced weight initialization strategies, the use of residual connections, and the implementation of activation functions like ReLU, which help mitigate the risk of gradient vanishing. These methods have proven effective in maintaining sufficient gradient flow throughout the network, thereby facilitating more effective and efficient training of deep neural models.

4.2 Methodology

This chapter presents the detailed training architecture and scheme for the state representation model. The architecture of the autoencoder, including the state representation model, is illustrated in Figure 4.1. The architecture comprises four stacks of convolutional layers, each followed by LeakyReLU activation functions. The choice of LeakyReLU over the standard ReLU is deliberate; while ReLU activation can lead to neurons 'dying' during training (i.e., outputting zero for all inputs), LeakyReLU mitigates this issue by allowing a small, non-zero gradient when the unit is inactive. This charactyeristic helps maintain a steady gradient flow, promoting more effective and consistent learning.

Each convolutional layer in the encoder utilizes a kernel size of $(3,\ 3)$, a stride of $(2,\ 2)$, and padding of $(1,\ 1)$. The LeakyReLU activation function employs a negative slope of 0.01. The encoder begins by transforming the input batch of $3 \times 64 \times 64$ image tensors into $32 \times 50 \times 50$ tensors through the first convolutional layer. The second layer further reduces these tensors to $64 \times 25 \times 25$, followed by the third layer, which converts them into $128 \times 13 \times 13$ tensors. The final convolutional layer produces $256 \times 6 \times 6$ tensors. After these four convolutional layers, the output is

passed through two linear layers that flatten and sequentially compress the data into the latent representation denoted as h.

The decoder is designed symmetrically opposite to the encoder, reconstructing the original input image from the latent representation. Each convolutional layer in the decoder mirrors its corresponding layer in the encoder, ensuring the transformation process is effectively reversed. In the last layer of the decoder, a sigmoid activation function is used. This exception is made because the sigmoid function maps the output to the range $[0,\ 1]$, which is appropriate for image pixel values and helps in producing more realistic reconstructed images.

To enhance the learning process and mitigate issues such as the vanishing gradient problem discussed in Chapter 4.1, skip connections were incorporated in the architecture. Skip connections, or residual connections, involve bypassing certain layers by routing the output of one layer as input to a subsequent layer further along in the network. These connections help preserve the information and gradients across the network layers, facilitating more efficient training and better model performance. In the state representation model, four skip connections were formed: the output of the encoder's first layer is connected to the decoder's last layer, the encoder's second layer output to to the decoder's third layer input, the encoder's third layer output to the decoder's second layer input, and the encoder's last layer output to the decoder's first layer input.

4.3 Results and Discussion

In this Chapter, the training results of the state representation model with ablation study on skip connections are discussed.

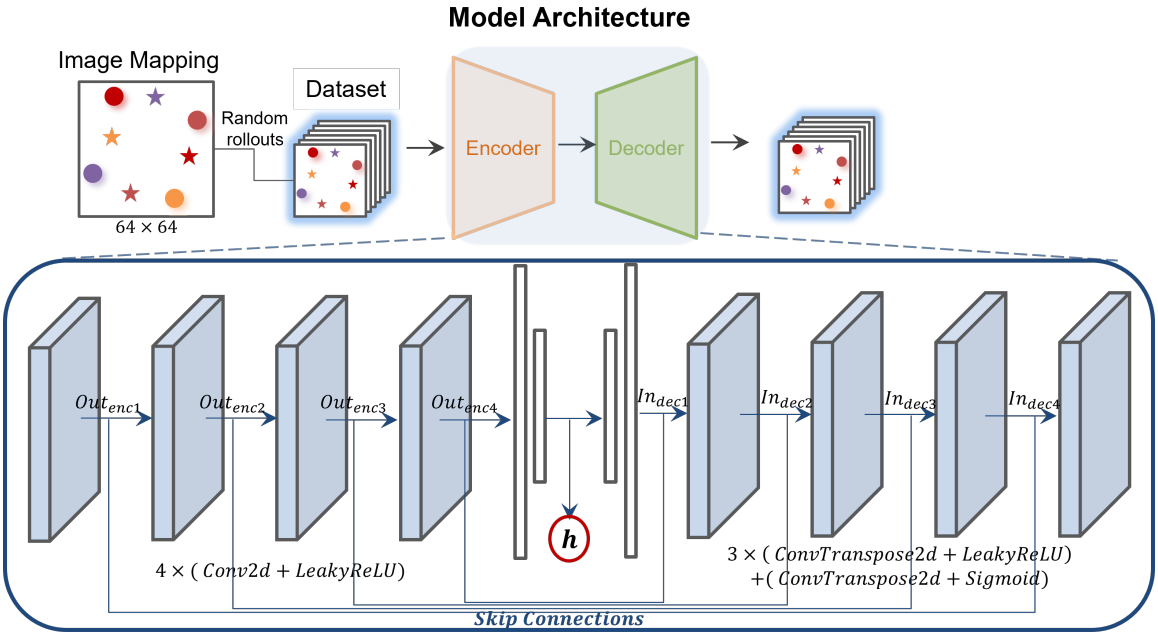


Figure 4.1: The model architecture of AE for training the state representation model of global image map. After the training the network, the encoder is extracted as the state representation model that abstracts the global map into global context h.

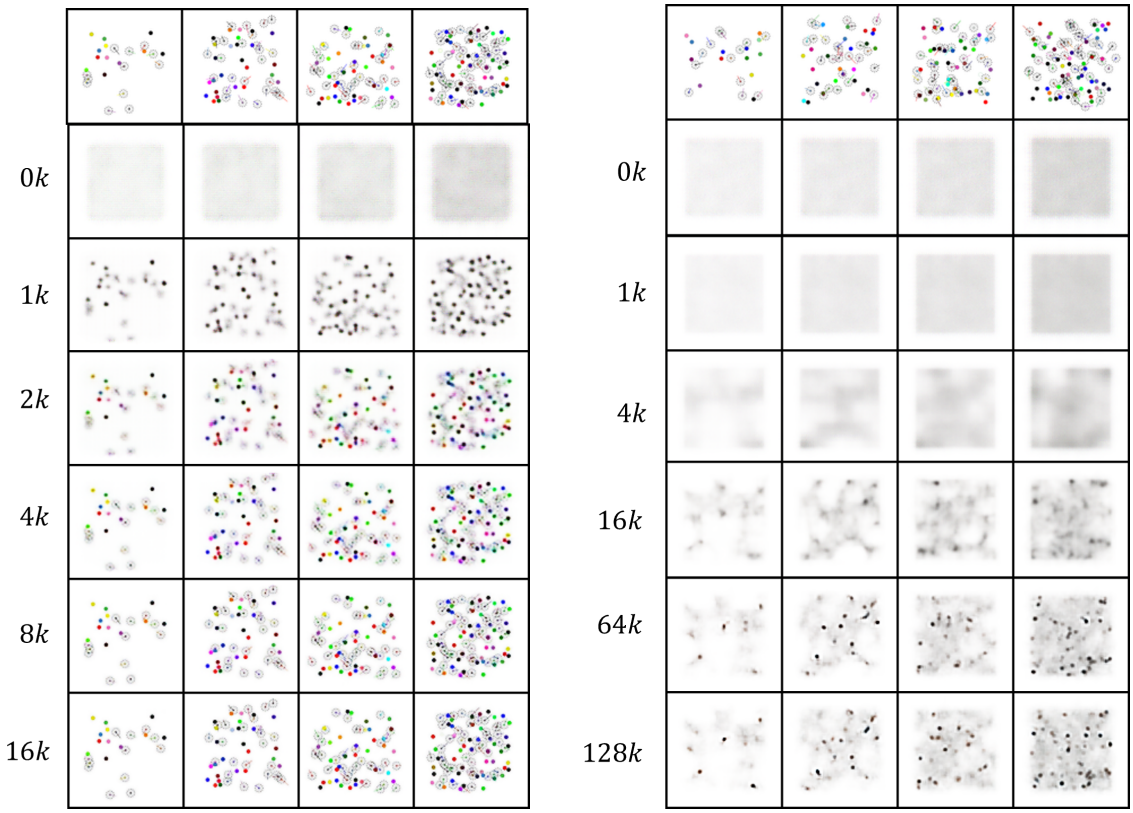
Ablation Study on Skip Connections

Fig 4.2 displays the results of reconstructed image samples from the validation dataset during the training steps. The results show that the inclusion of skip connections significantly enhances the quality of reconstructed images. Images reconstructed with skip connections retain much more detail compared to those without skip connections in the early training $16k$ steps. Even after $100k$ steps, the network trained without skip connections struggles to retain and reconstruct fine details, leading to blurred and less accurate images. This suggests that skip connections effectively mitigate the vanishing gradient problem, allowing the network to extract meaningful global context features from the input data.

Training Slope Analysis

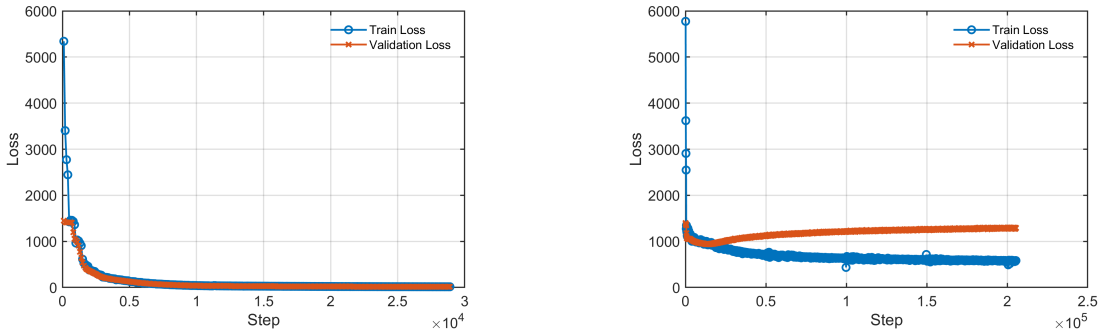
Fig 4.3 presents the training slopes for the state representation model, with the left plot corresponding to training with skip connections and the right corresponding to training without skip connections. According to the result, skip connections contribute to a more stable and faster convergence during training. The loss curves for both training and validation datasets indicates that the model with skip connections not only learns faster but also generalizes better to unseen data. Conversely, the model without skip connections exhibits higher loss values and overfitting, highlighting difficulties in converging to find meaningful global context. Moreover, the overfitting demonstrates that the network is extracting some biased features to training dataset.

The results demonstrate the advantages of incorporating skip connections in the state representation model's architecture. Skip connections facilitate better gradient flow, effectively addressing the vanishing gradient problem and enabling the network to learn features more effectively. This results in higher quality reconstructions and more stable training dynamics.



(a) With Skip Connections (b) Without Skip Connections

Figure 4.2: The reconstructed image samples from the validation dataset while training. The top row images of both 4.2a and 4.2b are the original images. The numbers listed at the leftmost column denotes the training steps when the AE reconstructed the original image.



(a) With Skip Connections (b) Without Skip Connections

Figure 4.3: The training slopes for while training the AE.

Chapter 5

Multi-Agent Proximal Policy Optimization for WM-MAPF

PPO is a ubiquitous on-policy DRL algorithm but is not utilized as many as off-policy DRL algorithms in multi-agent settings. This often stems from the belief that on-policy algorithms are significantly sample inefficient than off-policy methods in multi-agent settings. However, recent study from [118] demonstrated the potential of PPO-based multi-agent algorithms by its strong performance from experimental results in complicated multi-agent environments. Moreover, it further surpass competitive off-policy methods in sample efficiency. The environment adopted in this thesis for solving MAPF problem was VMAS, which is a vectorized fully-differentiable simulator with 2-dimensional physics engine. This chapter provides the theoretical background of MAPPO and a detailed methodology of *World Model* based MAPPO algorithm employed in VMAS environment.

5.1 Theoretical Background of Multi-Agent Proximal Policy Optimization

Natural Policy Gradient

The challenge of AC method is how to select the appropriate learning rate α. Even though the selected α size exhibits a good value for one update, it can cause an overshoot for the next update and end up with a worse policy.

Natural Policy Gradient (NPG) [46] addresses this issue by including a constraint on policy changes during the update. It updates the policy network putting a cap ϵ on KL-divergence between policy changes:

$$D_{KL}(\pi_\omega \| \pi_{\omega+\Delta\omega}) = \sum_{x \in \mathcal{X}} \pi_\omega(x) \cdot \log(\frac{\pi_\omega(x)}{\pi_{\omega+\Delta\omega}(x)}), \tag{5.1}$$

$$\Delta\omega^* = \text{argmax}_{D_{KL}(\pi_\omega \| \pi_{\omega+\Delta\omega}) \leq \epsilon} J(\omega + \Delta\omega). \tag{5.2}$$

Eq. 5.2 can be simplified by Lagrangian relaxation which transform the divergence constraint into a penalty, yielding an expression that is easier to solve:

$$\Delta\omega^* = \text{argmax}_{\Delta\omega} \Big(J(\omega + \Delta\omega) - \lambda(D_{KL}(\pi_\omega \| \pi_{\omega+\Delta\omega}) - \epsilon) \Big). \tag{5.3}$$

With first-order and second-order Taylor expansion on the objective function and KL-divergence, Eq. 5.3 can be approximated as follows:

$$\begin{aligned} \Delta\omega^* \simeq \text{argmax}_{\Delta\omega} \Big(& J(\omega_{old}) + \nabla_\omega J(\omega)|_{\omega=\omega_{old}} \cdot \Delta\omega \\ & -\frac{1}{2}\lambda(\Delta\omega^T \nabla^2_\omega D_{KL}(\pi_{\omega_{old}} || \pi_\omega)|_{\omega=\omega_{old}} \Delta\omega) + \lambda\epsilon \Big) \end{aligned} \tag{5.4}$$

The solution of Eq. 5.4 can be derives as

$$\Delta\omega = \sqrt{\frac{2\epsilon}{\nabla J(\omega)^T \cdot F(\omega)^{-1} \cdot \nabla J(\omega)}}\tilde{\nabla} J(\omega), \tag{5.5}$$

where $F(\omega)$ is the Fisher information matrix, which describes the curvature of KL-divergence:

$$F(\omega) = \nabla^2_\omega D_{KL}(\pi_\omega \| \pi_{\omega+\Delta\omega}), \tag{5.6}$$

and $\tilde{\nabla} J(\omega)$ is the NPG, which is the gradient corrected for the curvature:

$$\tilde{\nabla} J(\omega) = F(\omega)^{-1} \nabla J(\omega). \tag{5.7}$$

NPG still requires a lot of approximations which may misrepresent the actual objective function and curvature of KL-divergence. Besides, the Fisher information matrix $F(\omega)$ often requires a huge memory for neural network parameters.

TRPO

TRPO [78] tackles these challenges. Rather than putting a constraint on policy changes like NPG, TRPO utilizes the concept of advantage function to describe the difference in expected rewards between the original policy and updated policy by taking the expected reward of the original policy and adding the expected advantage of the new policy:

$$\begin{aligned}
\nabla J(\omega) &= \sum_{s\in S} \rho_{\pi_{\omega+\Delta\omega}}(s) \sum_{a\in A} \nabla_\omega \log \pi_{\omega+\Delta\omega}(a|s) \cdot \mathrm{A}_\omega(s,a) \\
&\simeq \sum_{s\in S} \rho_{\pi_\omega}(s) \sum_{a\in A} \nabla_\omega \log \pi_{\omega+\Delta\omega}(a|s) \cdot \mathrm{A}_\omega(s,a) \\
&\simeq \mathbb{E}_{s\sim\rho_{\pi_\omega}} \frac{\pi_{\omega+\Delta\omega}(a|s)}{\pi_\omega(a|s)} \mathrm{A}_\omega(s,a) = \mathcal{L}_\omega(\pi_{\omega+\Delta\omega}),
\end{aligned} \tag{5.8}$$

where ρ_{π_ω} denotes the state distribution, and $\mathcal{L}_\omega(\pi_{\omega+\Delta\omega})$ denotes the surrogate advantage which describes the quality of the updated policy relative to the original one. The approximation error can be expressed in terms of the worst-case KL-divergence between both policies:

$$\nabla J(\pi_{\omega+\Delta\omega}) \geq \mathcal{L}_\omega(\pi_{\omega+\Delta\omega}) - C \cdot D_{KL}^{max}(\pi_\omega || \pi_{\omega+\Delta\omega}), \tag{5.9}$$

where C is the penalty that can be derived analytically. In Ineq. 5.9, we are guaranteed to improve the policy by maximizing the lower bound. Unlike NPG, TRPO enforces improvement in policy updates when it is coupled with conjugate gradient approximation and line search algorithm.

PPO

PPO is also an advanced policy-gradient method and is an on-policy RL algorithm that learns from the actions taken within the current policy. It has been demonstrated that it is empirically competitive to TRPO but considerably simpler to implement. Instead of imposing a hard constraint, PPO clips the objective function:

$$\mathcal{L}_{\omega_{old}}^{CLIP}(\omega) = \mathbb{E}_{\tau\sim\pi_{\omega_{old}}}\Big[\sum_{t=0}^{T}\big[\min(r_t(\omega)\cdot \mathrm{A}_{\omega_{old}}^{(t)}, clip(r_t(\omega), 1-\epsilon, 1+\epsilon)\cdot \mathrm{A}_{\omega_{old}}^{(t)})\big]\Big], \tag{5.10}$$

where $r_t(\omega) = \pi_\omega(a_t|s_t)/\pi_{\omega_{old}}(a_t|s_t)$. PPO basically introduces the idea of importance sampling to evaluate a new policy π_ω with samples collected from the original policy $\pi_{\omega_{old}}$. Then, after a certain number of iterations, it updates the original policy to the new policy:

$$\pi_{\omega_{old}}(a_t|s_t) \leftarrow \pi_\omega(a_t|s_t). \tag{5.11}$$

Algorithm 1 is the pseudocode procedure of PPO. Although PPO lacks some theoretical guarantees and mathematical finesse compared to TRPO or NPG, it strikes the right balance between speed, caution, and usability. Besides, within CTDE paradigm where centralized value function inputs and policy parameter-sharing agents, the PPO framework can be extended to MAPPO with improved performance.

Algorithm 1 PPO with Clipped Objective

Require: $\theta_0, \omega_0, \epsilon$

▷ initial value function parameters θ_0, initial policy parameters ω_0, clipping threshold ϵ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories τ_k from policy $\pi_k = \pi(\omega_k)$.

Compute expected returns $\mathcal{R}_t$ and generalized advantage estimates $\hat{\mathrm{A}}_{\omega_k^{(t)}}$ using τ_k based on the current value function $\mathcal{V}_{\theta_k}$.

Compute value function update

$$\theta_{k+1} = \mathrm{argmin}_\theta \mathbb{E}_{\tau_k}\Big[\sum_{t=0}^{T}\big(\mathcal{V}_\theta(\tau_t(s)) - \mathcal{R}_t\big)^2\Big] \text{ , and}$$

Compute policy update

$$\omega_{k+1} = \mathrm{argmax}_\omega \mathbb{E}_{\tau_k}\left[\sum_{t=0}^{T}\left[\min\Big(r_t(\omega)\cdot \mathrm{A}^{(t)}_{\omega_k}, clip\big(r_t(\omega), 1-\epsilon, 1+\epsilon\big)\cdot \mathrm{A}^{(t)}_{\omega_k}\Big)\right]\right],$$

where

$$r_t(\omega) = \frac{\pi_\omega(\tau_t(a)|\tau_t(s))}{\pi_{\omega_k}(\tau_t(a)|\tau_t(s))},$$

by taking K steps of minibatch SGD (via Adam).

end for

MAPPO

Algorithm 2 shows the details of the pseudocode of MAPPO. In MAPPO, the critic-network parameters θ and actor-network parameters ω are shared amongst all agents, but each agent can have its own pair of actor-critic networks. The critic-network V_θ performs the mapping between the global state and the return: $S \to \mathbb{R}$. The actor-network π_ω, on the other hand, maps agent observation $o_t^{(a)}$ to a distribution over actions in action spaces. In discrete space, the network outputs a categorical

distribution among the action spaces. In continuous space, the network outputs the mean and standard deviation vectors of a Multivariate Gaussian Distribution so that an action can be sampled from it.

The critic-network is trained to minimize the loss function

$$\mathcal{L}(\theta) = \frac{1}{B \cdot n} \sum_{i=1}^{B} \sum_{k=1}^{n} \Bigg[\max \Bigg[\Big(V_\theta(s_i^{(k)}) - \hat{R}_i \Big)^2, \Big(\text{clip}(V_\theta(s_i^{(k)}), V_{\theta_{old}}(s_i^{(k)}) - \epsilon, V_{\theta_{old}}(s_i^{(k)}) + \epsilon) - \hat{R}_i \Big)^2 \Bigg] \Bigg] \tag{5.12}$$

where B refers to the batch size, n refers to the number of agents, and $\hat{R}_i$ is the discounted reward-to-go. The actor-network is trained to maximize the objective function

$$\mathcal{L}(\omega) = \frac{1}{B \cdot n} \sum_{i=1}^{B} \sum_{k=1}^{n} \left[\min \left(r_{\omega,i}^{(k)} \cdot \hat{A}_i^{(k)}, \left(\text{clip}(r_{\omega,i}^{(k)}, 1-\epsilon, 1+\epsilon) \cdot \hat{A}_i^{(k)} \right) \right) + \sigma \cdot S\Big(\pi_\omega(o_i^{(k)}) \Big) \right] \tag{5.13}$$

where importance sampling for off-policy corrections $r_{\omega,i}^{(k)}$ can be derived

$$r_{\omega,i}^{(k)} = \frac{\pi_\omega(a_i^{(k)} | o_i^{(k)})}{\pi_{\omega_{old}}(a_i^{(k)} | o_i^{(k)})}. \tag{5.14}$$

$\hat{A}_i^{(k)}$ is computed using the GAE method, S is the policy entropy, and σ is the entropy coefficient hyperparameter.

Algorithm 2 Recurrent-MAPPO

Require: $\theta, \omega \triangleright$ initial parameters for critic V: θ, initial parameters for policy π: ω
  Set learning rate α
  while $step \leq step_{\max}$ **do**
    set data buffer $D = \{\}$
    for $i = 1$ to $batch\_size$ **do**
      $\tau = []$ empty list
      initialize $h_{0,\pi}^{(1)}, \dots h_{0,\pi}^{(n)}$ actor RNN states
      initialize $h_{0,V}^{(1)}, \dots h_{0,V}^{(n)}$ critic RNN states
      for $t = 1$ to T **do**
        for all agents a **do**
          $p_t^{(a)}, h_{t,\pi}^{(a)} = \pi(o_t^{(a)}, h_{t-1,\pi}^{(a)}; \theta)$
          $u_t^{(a)} \sim p_t^{(a)}$
          $v_t^{(a)}, h_{t,V}^{(a)} = V(s_t^{(a)}, h_{t-1,V}^{(a)}; \omega)$
        end for
        Execute actions $\boldsymbol{u}_t$, observe $r_t, s_{t+1}, \boldsymbol{o}_{t+1}$
        $\tau += [s_t, \boldsymbol{o}_t, \boldsymbol{h}_{t,\pi}, \boldsymbol{h}_{t,V}, \boldsymbol{u}_t, r_t, s_{t+1}, \boldsymbol{o}_{t+1}]$
      end for
      Compute advantage estimate $\hat{A}$ via GAE on τ
      Compute reward-to-go $\hat{R}$ on τ
      Split trajectory τ into chunks of length L
      for $l = 0, 1, \dots, T//\mathrm{L}$ **do**
        $D = D \cup (\tau[l : l + T, \hat{A}[l : l + \mathrm{L}], \hat{R}[l : l + \mathrm{L}]])$
      end for
    end for
    for mini-batch $k = 1, \dots, K$ **do**
      $b \leftarrow$ random mini-batch from D with all agent data
      for each data chunk c in the mini-batch b **do**
        update RNN hidden states for π and V from first hidden state in data chunk
      end for
    end for
    Adam update θ on $\mathcal{L}(\theta)$ with data b
    Adam update ω on $\mathcal{L}(\omega)$ with data b
  end while

5.2 WM-MAPPO in VMAS Navigation Scenario

As PPO is an on-policy algorithm, each learning iteration involves two phases: the data sampling phase and the training phase. During the data sampling phase, rollouts are collected from the interactions of all agents with the environment using the current policies π_t. In the training phase, all collected rollouts are immediately fed to the training process and consumed for a certain number of epochs to update both the value function network and policy network. This results in updated policies, which are then used in the subsequent learning step.

In the sampling phase, the state representation model is introduced into the environment as a centralized communication block. This block provides the global context from global image map at each timestep. The state representation model is extracted from the encoder of pre-trained autoencoder as discussed in Chapter 4.3. Therefore, each agent preceives not only its local observation from the environment but also the global context from the communication block at each timestep. Agents then generate actions based on their observations and the global context to interact with the environment and receive rewards. The rollouts continue until the interaction data accumulate to the batch size required for training. The batch consists of tuples in the form (state, action, reward, next state, done), representing the transition from one state to another under a specific action and the resultant reward. Once the batch of sampled data is acquired, the phase give turns to the training phase.

The batch of sampled data is stored in a replay buffer. Replay buffer is a data storage mechanism typically utilized in off-policy DRL algorithms to store the agent's experience over time and randomly sample mini-batches for training network parameters. This random sampling helps break the correlation between consecutive experiences, thereby stabilizing the learning process and reflecting the overall dynamics of the environment. In this work, however, the replay buffer is refilled at each learning

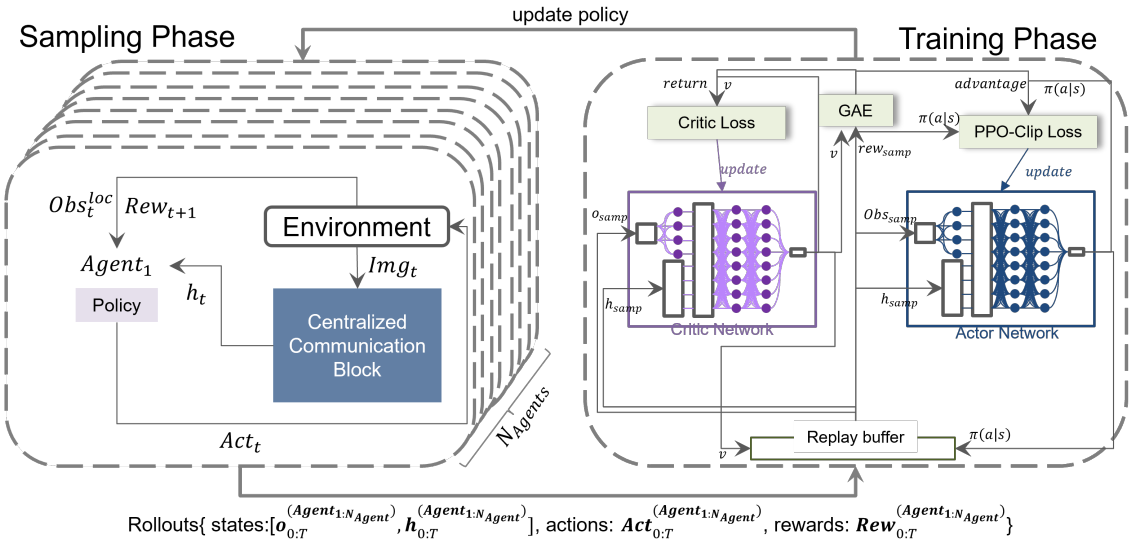


Figure 5.1: The on-policy learning phases of WM-MAPPO framework.

iteration with newly collected rollout data and consumes it for a certain number of epochs, becoming empty after the training phase ends. The learning phases of the on-policy DRL algorithm used in WM-MAPPO is depicted in Figure 5.1

In the training phase, a critic network is used to estimate wheter the actions taken by the policy network were effective. The critic can be adapted to approximate different values depending on the specific objective. In Independent-PPO (IPPO), the critic estimates the mean discounted return expected from an individual agent's local observation. The critic loss compares the actual return, computed from the batch, to the return estimated by the critic. This comparison helps determine the advantage of the action taken and guides the policy optimization. Consequently, the training process is conducted in a decentralized manner, as both the critic and the policy network require only local information to compute their outputs. Conversely, in MAPPO, the critic network takes the concatenation of the agents' observation as input and estimates the aggregated mean discounted return from all agents. This approach facilitates centralized training and decentralized execution, addressing the non-stationary caused by multiple agents by incorporating global context. However,

it may be impacted by the large input space. Both methods assume that the policy network is shared among agents, enabling them to benefit from each other's experiences and leading to faster training.

The training phase of WM-MAPPO is similar to IPPO in that the critic estimates the value of individual agents' specific states. However, the abstracted global context provided by the centralized communication block includes part of the agents' states, addressing both the non-stationarity issue and the challenge of the large input space. The critic network outputs value estimates for the current and next states using the discounted return V_θ. Then, it is updated by the critic loss function, which clips the mean squared error between the target value $\hat{R}$ derived from GAE using the equation 5.12. The Adam optimizer is employed for backpropagation of the gradients through the network parameters.

The actor network is updated using the equation 5.13 and equation 5.14. Given that the action space is continuous, the agents utilizes a stochastic policy to facilitate exploration. Instead of outputting a single value corresponding to the action taken, the actor network outputs the parameters of distributions. Additionally, a Tanh-Normal distribution is used to respect the action space boundaries. Each agent's action is represented by 2-dimensional independent normal distributions, with the distribution parameters being outputted from the actor network for each distribution:

$$\mathrm{Net}_\omega(o_t^{(a)}, h_t) = \boldsymbol{\mu}_t^{(a)},\ \boldsymbol{\sigma}_t^{(a)}, \tag{5.15}$$

where Net_ω denotes the policy network, $o_t^{(a)}$ represents agent a's local observation at timestep t, and h_t denotes the global context delivered by centralized communication block at timestep t. Here, $\boldsymbol{\mu}_t^{(a)}$ and $\boldsymbol{\sigma}_t^{(a)}$ denote the mean and standard deviation of

the action distributions, respectively. The policy action is then derived as follows:

$$(u_{x,t}^{(a)},\ u_{y,t}^{(a)}) = \pi_\omega(o_t^{(a)},\ h_t) = \text{TanhNorm}(\boldsymbol{\mu}_t^{(a)},\ \boldsymbol{\sigma}_t^{(a)}) \tag{5.16}$$

where

$$\text{TanhNorm}(\mu,\ \sigma) = \frac{e^{\rho} - e^{-\rho}}{e^{\rho} + e^{\rho}}, \tag{5.17}$$

$$\rho \sim \varphi(\mu,\ \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\rho-\mu)^2}{2\sigma^2}}. \tag{5.18}$$

The network architecture of both the actor and critic networks is illustrated in Figure 5.2. Both networks share the same structure except for the final layer. The initial layers consist of a combination of a fully connected network and LeakyReLU activation function. The layer receives the local observation of an agent and produce an output with the same dimensionality as the global context h. This output is then concatenated with h and fed into the second layer. The second and third layers output values with the same dimensionality as the input, using LeakyReLU activation functions. A residual connection is introduced before the activation block of the third layer, connecting it to the input of second layer. This residual connection enables faster training, as discussed in Chapter 4. The critic network subsequently outputs the value function through an additional fully connected layer. In contrast, the actor network diverges the input at the last layer into two streams: one that outputs the mean value and another that outputs the logarithmic standard deviation, which is then transformed into the standard deviation.

The hyperparameter details of training WM-MAPPO are listed in Table 5.1. ϵ_{clip} refers to the value-clipping threshold for PPO loss, γ to discount factor, λ to GAE coefficient, and ϵ_{etp} to entropy multiplier for computing the total loss. Following the recommendations in [118], the networks are trained with 5 epochs per update to enhance the stability of policy and value learning. Additionally, the training data is

split into two mini-batches to improve practical performance by utilizing more data for gradient estimation, as suggested in [44].

Table 5.1: Hyperparameters for training MAPPO

Hyperparameters	***Value***
ϵ_{clip}	0.2
γ	0.9
λ	0.9
ϵ_{etp}	0.0001

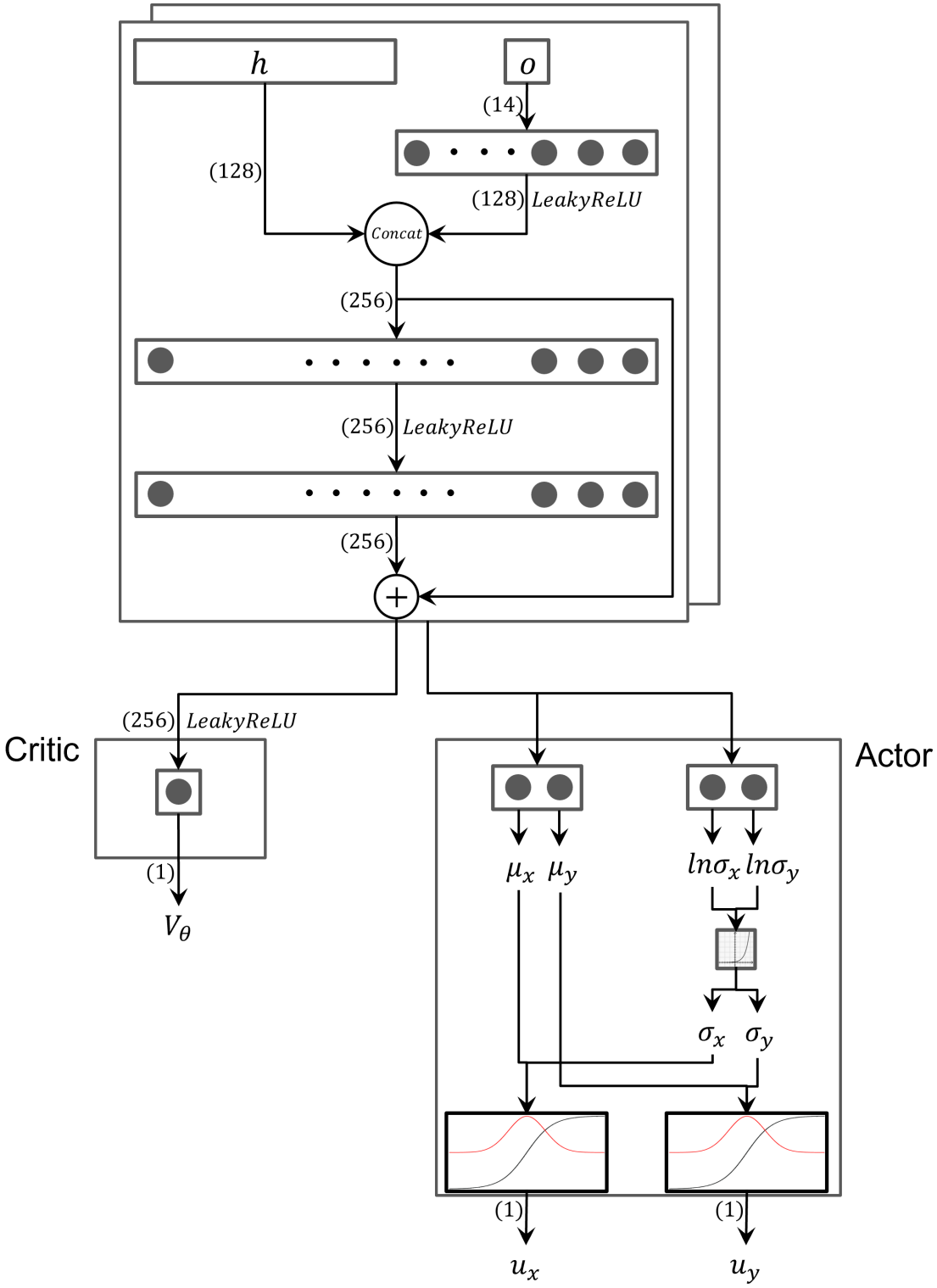


Figure 5.2: The network architecture of actor and critic networks in WM-MAPPO. In the final layer of actor, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are fed into tanh-normal distribution to sample u_x and u_y.

5.3 Experiment Results

This chapter presents an empirical study to evaluate the effectiveness of the proposed approach in MAPF. The performance of IPPO, MAPPO, and WM-MAPPO was compared under identical conditions. Figure 5.3 illustrates the training results, showing the converged mean episode return per agent across environments with varying numbers of agents. All frameworks were trained with 128 episodes for $20, 40, 60, 80$, and 100 agents. The results indicate that the mean episode reward tends to decline when the number of agents exceeds 80. This decline is attributed to the reward structure of the VMAS environment, where team rewards are granted when all agents successfully reach their goals. The reduction in reward with a larger number of agents suggests that congestion, along with an increased likelihood of collisions, may prevent the entire team from reaching their goals within the constrained map. Nevertheless, The training result demonstrate that the proposed approach outperforms both IPPO and MAPPO, indicating that the agents are able to effectively utilize the global context for improved decision-making.

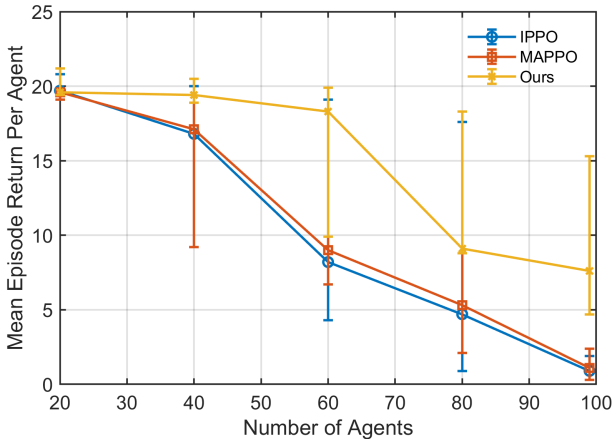


Figure 5.3: the plot of converged mean episode return per agent while training the frameworks with varying number of agents.

Following the training of each framework, the policies trained in environments with

20 agents were subsequently tested in scenarios with varying numbers of agents using 128 unseen episodes. Figure 5.4 presents the mean success rate for each framework across different numbers of agents. While the mean success rate of IPPO and MAPPO decreases significantly when the number of agents exceeds 80, the proposed WM-MAPPO model maintains over 80% of success rate. This result highlights that our proposed framework also enhanced the generalization capability of the policy.

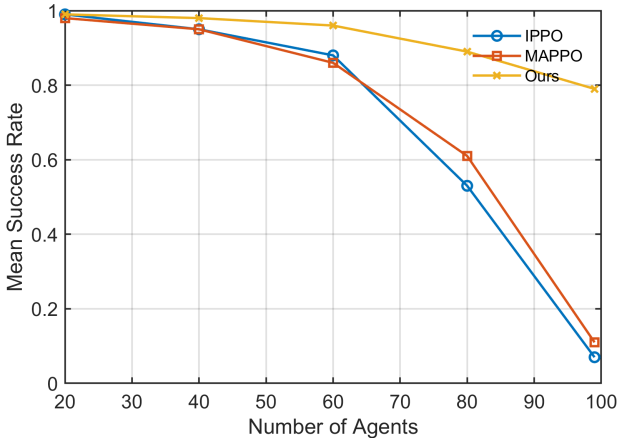


Figure 5.4: the plot of mean success rate of frameworks with varying number of agents in test environments.

Chapter 6

Conclusions and Future Work

6.1 Summary

This thesis contributes to the advancement of MAPF with state representation model playing as a centralized communication block in MAPPO framework, achieving outperforming results than the baseline PPO policies. Specifically, a WM-MAPPO framework is proposed, where a WM is introduced to extract the spatial context of the global map in the MAPF environment and provide the global context to the agents. The empirical studies between three PPO-based MAPF frameworks demonstrates that the proposed framework could sufficiently provide the global context, fostering improved generalization capability as well as better policies.

The introduction of a centralized communication block to multi agent decentralized system represents a significant advancement with substantial implications for real-world applications. By facilitating the seamless sharing of global context information among agents, this centralized mechanism addresses common challenges such as non-stationarity and limited observability, which are prevalent in practical settings. This improvement allows for more synchronized and informed decision making, enabling

agents to operate more efficiently and effectively in dynamic environments. This innovation is particulary valuable in fields such as robotics, logistics, and autonomous vehicle coordination, where the ability to quickly adapt to changing conditions and coordinate actions across multiple agents can lead to significant operational efficiencies and safety improvements.

The proposed framework still shows that there is a limitation when applied to a congested environment. However, it builds the foundation of future work and has the potential to address the issue and to provide further improvements. Firstly, the WM can provide global information not only to the critic but also to the actor as it abstracts the high-dimensional global information to a size-invariant low-dimensional latent vector. This would allow the agents' policy network to have a compact memory size with low computational costs for the policy network. In addition, it reduces both non-stationarity and partial observability of the agents in the MAPF environment, without formulating a communication algorithm between the agents and still maintaining decentralized execution.

6.2 Limitations and Future Directions

The results of the suggested framework reveal certain limitations that suggest potential avenues for future research:

- **Potential Development on State Representation Model:** In this research, AE was trained to capture only a single frame of the global map to extract contextual features. However, spatio-temporal information could be extracted by adopting sequence models or attention-based models for the state representation model. Additionally, the centralized communication block, in this thesis, transmits the global context unilaterally, without receiving signals from

agents. Future work could develop the centralized communication block to send agent-specific global contexts, which would substantially reduce the state space and enhance the learning process.

- **Uncertainty and Processing of Sensory data:** The environmental setup used in this thesis is simulation-based, where the LIDAR sensor readings are assumed to be flawless. However, in real-world applications, sensory data often includes noise. Furthermore, the agents in this thesis received only 14 sensory readings, which may provide too low resolution for practical applications. This issue can be addressed by introducing uncertainty into the network models or integrating intelligent processing mechanisms to handle noisy data effectively.

- **Sim-to-Real Applications:** Transferring trained DRL policies from simulation environments to reality is a crucial step for the practical utility of MAPF algorithms. While the agent policies in the proposed framework directly produce actions from sensory data and global context in an end-to-end manner, bridging the gap between simulation and reality presents a significant challenge. This transition often requires addressing discrepancies between simulated and real-world conditions, such as differences in sensor noise, environmental dynamics, and physical interactions. Future work could focus on developing robust transfer learning techniques, domain adaptation strategies, and improved sensor models to ensure that the policies trained in simulations perform effectively in real-world scenarios.

Bibliography

[1] Nafis Ahmed, Chaitali J Pawase, and KyungHi Chang. Distributed 3-d path planning for multi-uavs with full area surveillance based on particle swarm optimization. *Applied Sciences*, 11(8):3417, 2021.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998.

[4] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[6] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009.

[7] Frodo Kin Sun Chan, Yan Nei Law, Bonny Lu, Tom Chick, Edmond Shiao Bun Lai, and Ming Ge. Multi-agent pathfinding for deadlock avoidance on rotational movements. In *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 765–770. IEEE, 2022.

[8] Baiming Chen, Mengdi Xu, Zuxin Liu, Liang Li, and Ding Zhao. Delay-aware multi-agent reinforcement learning for cooperative and competitive environments. *arXiv preprint arXiv:2005.05441*, 2020.

[9] Lin Chen, Yaonan Wang, Zhiqiang Miao, Yang Mo, Mingtao Feng, and Zhen Zhou. Multi-agent path finding using imitation-reinforcement learning with transformer. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 445–450. IEEE, 2022.

[10] Lin Chen, Yaonan Wang, Yang Mo, Zhiqiang Miao, Hesheng Wang, Mingtao Feng, and Sifei Wang. Multi-agent path finding using deep reinforcement learning coupled with hot supervision contrastive loss. *IEEE Transactions on Industrial Electronics*, 2022.

[11] Jaehoon Chung, Jamil Fayyad, Younes Al Younes, and Homayoun Najjaran. Learning team-based navigation: a review of deep reinforcement learning techniques for multi-agent pathfinding. *Artificial Intelligence Review*, 57(2):41, 2024.

[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[13] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[14] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

[15] Pradipta Kumar Das, Himansu Sekhar Behera, Swagatam Das, Hrudaya Kumar Tripathy, Bijaya K Panigrahi, and SK Pradhan. A hybrid improved pso-dv algorithm for multi-robot path planning in a clutter environment. *Neurocomputing*, 207:735–753, 2016.

[16] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[17] Stepan Dergachev and Konstantin Yakovlev. Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1489–1494. IEEE, 2021.

[18] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

[19] Esra Erdem, Doga Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 290–296, 2013.

[20] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[21] Michael Everett, Yu Fan Chen, and Jonathan P How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021.

[22] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7):856–892, 2020.

[23] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[24] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8:325–344, 2000.

[25] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3582–3589, 2019.

[26] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan Sturtevant, Robert C Holte, and Jonathan Schaeffer. Enhanced partial expansion a. *Journal of Artificial Intelligence Research*, 50:141–187, 2014.

[27] Eric J Griffith and Srinivas Akella. Coordinating multiple droplets in planar array digital microfluidic systems. *The International Journal of Robotics Research*, 24(11):933–949, 2005.

[28] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2022.

[29] Huifeng Guan, Yuan Gao, Min Zhao, Yong Yang, Fuqin Deng, and Tin Lun Lam. Ab-mapper: Attention and bicnet based multi-agent path planning for dynamic environment. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13799–13806. IEEE, 2022.

[30] Siyu Guo, Xiuguo Zhang, Yisong Zheng, and Yiquan Du. An autonomous path planning model for unmanned ships based on deep reinforcement learning. *Sensors*, 20(2):426, 2020.

[31] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[32] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[33] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[34] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[35] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[36] Dan Halperin, Jean-Claude Latombe, and Randall H Wilson. A general framework for assembly planning: The motion space approach. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 9–18, 1998.

[37] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[38] Chengyang He, Tianze Yang, Tanishq Duhan, Yutong Wang, and Guillaume Sartoretti. Alpha: Attention-based long-horizon pathfinding in highly-structured areas. *arXiv preprint arXiv:2310.08350*, 2023.

[39] Zichen He, Lu Dong, Changyin Sun, and Jiawei Wang. Asynchronous multithreading reinforcement-learning-based path planning and tracking for unmanned underwater vehicle. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(5):2757–2769, 2021.

[40] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[41] Hongtao Hu, Xurui Yang, Shichang Xiao, and Feiyang Wang. Anti-conflict agv path planning in automated container terminals based on multi-agent reinforcement learning. *International Journal of Production Research*, pages 1–16, 2023.

[42] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments

via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020.

[43] Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, 52:2191–2233, 2019.

[44] Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients, 2020.

[45] James S Jennings, Greg Whelan, and William F Evans. Cooperative search and rescue with a team of mobile robots. In *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*, pages 193–200. IEEE, 1997.

[46] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

[47] Navroop Kaur and Harjot Kaur. A multi-agent based evacuation planning for disaster management: a narrative review. *Archives of Computational Methods in Engineering*, 29(6):4085–4113, 2022.

[48] Justin Kottinger, Shaull Almagor, and Morteza Lahijanian. Conflict-based search for multi-robot motion planning with kinodynamic constraints. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13494–13499. IEEE, 2022.

[49] Matthew Lai. Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549*, 2015.

[50] Haodong Li, Tao Zhao, and Songyi Dian. Prioritized planning algorithm for multi-robot collision avoidance based on artificial untraversable vertex. *Applied Intelligence*, 52(1):429–451, 2022.

[51] Wenhao Li, Hongjun Chen, Bo Jin, Wenzhe Tan, Hongyuan Zha, and Xiangfeng Wang. Multi-agent path finding with prioritized communication learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10695–10701. IEEE, 2022.

[52] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[53] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[54] Zuxin Liu, Baiming Chen, Hongyi Zhou, Guru Koushik, Martial Hebert, and Ding Zhao. Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11748–11754. IEEE, 2020.

[55] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[56] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[57] Hang Ma, Craig Tovey, Guni Sharon, TK Kumar, and Sven Koenig. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[58] Ziyuan Ma, Yudong Luo, and Hang Ma. Distributed heuristic multi-agent path finding with communication. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8699–8705. IEEE, 2021.

[59] Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters*, 7(2):1455–1462, 2021.

[60] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[61] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[62] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. A framework for reinforcement learning and planning. *arXiv preprint arXiv:2006.15009*, 127, 2020.

[63] Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. A unifying framework for reinforcement learning and planning. *Frontiers in Artificial Intelligence*, 5:908353, 2022.

[64] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[65] Milad Nazarahari, Esmaeel Khanmirza, and Samira Doostie. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, 115:106–120, 2019.

[66] Keisuke Okumura, François Bonnet, Yasumasa Tamura, and Xavier Défago. Offline time-independent multiagent path planning. *IEEE Transactions on Robotics*, 2023.

[67] Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022.

[68] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[69] Sameera Poduri and Gaurav S Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 165–171. IEEE, 2004.

[70] Han Qie, Dianxi Shi, Tianlong Shen, Xinhai Xu, Yuan Li, and Liujing Wang. Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE access*, 7:146264–146272, 2019.

[71] Hong Qu, Ke Xing, and Takacs Alexander. An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. *Neurocomputing*, 120:509–517, 2013.

[72] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

[73] Abhijeet Ravankar, Ankit A Ravankar, Yukinori Kobayashi, and Takanori Emaru. Symbiotic navigation in multi-robot systems with remote obstacle knowledge sharing. *Sensors*, 17(7):1581, 2017.

[74] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

[75] Daniela Rus, Bruce Donald, and Jim Jennings. Moving furniture with teams of autonomous robots. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 235–242. IEEE, 1995.

[76] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[77] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[78] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[79] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[81] Adarsh Sehgal, Hung La, Sushil Louis, and Hai Nguyen. Deep reinforcement learning using genetic algorithm for parameter optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 596–601. IEEE, 2019.

[82] Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.

[83] Gursel Serpen and Chao Dou. Automated robotic parking systems: real-time, concurrent and multi-robot path planning in dynamic environments. *Applied Intelligence*, 42:231–251, 2015.

[84] Amir Seyyedabbasi and Farzad Kiani. Map-aco: An efficient protocol for multi-agent pathfinding in real-time wsn and decentralized iot systems. *Microprocessors and Microsystems*, 79:103325, 2020.

[85] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219:40–66, 2015.

[86] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence*, 195:470–495, 2013.

[87] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[88] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[89] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[90] David Šišlák, Přemysl Volf, and Michal Pěchouček. Agent-based cooperative decentralized airplane-collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):36–46, 2010.

[91] Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr I Panov. When to switch: Planning and learning for partially observable multi-agent pathfinding. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[92] Kyunghwan Son, Sungsoo Ahn, Roben Delos Reyes, Jinwoo Shin, and Yung Yi. Qtran++: improved value transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2006.12010*, 2020.

[93] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

[94] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 173–178, 2010.

[95] Roni Stern. Multi-agent path finding–an overview. *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pages 96–115, 2019.

[96] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[97] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[98] Biwei Tang, Kui Xiang, Muye Pang, and Zhu Zhanxia. Multi-robot path planning using an improved self-adaptive particle swarm optimization. *International Journal of Advanced Robotic Systems*, 17(5):1729881420936154, 2020.

[99] Tadahiro Taniguchi, Shingo Murata, Masahiro Suzuki, Dimitri Ognibene, Pablo Lanillos, Emre Ugur, Lorenzo Jamone, Tomoaki Nakamura, Alejandra Ciria, Bruno Lara, et al. World models and predictive coding for cognitive and developmental robotics: frontiers and challenges. *Advanced Robotics*, pages 1–27, 2023.

[100] Alexandre Trudeau and Christopher M Clark. Multi-robot path planning via genetic programming. *arXiv preprint arXiv:1912.09503*, 2019.

[101] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[102] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[103] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial intelligence*, 219:1–24, 2015.

[104] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.

[105] Di Wang, Hongbin Deng, and Zhenhua Pan. Mrcdrl: Multi-robot coordination with deep reinforcement learning. *Neurocomputing*, 406:68–76, 2020.

[106] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[107] KC Wang and Adi Botea. Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. *Journal of Artificial Intelligence Research*, 42:55–90, 2011.

[108] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. *arXiv preprint arXiv:2309.09777*, 2023.

[109] Xihuai Wang, Zhicheng Zhang, and Weinan Zhang. Model-based multi-agent reinforcement learning: Recent progress and prospects. *arXiv preprint arXiv:2203.10603*, 2022.

[110] Yubing Wang, Peng Bai, Xiaolong Liang, Weijia Wang, Jiaqiang Zhang, and Qixi Fu. Reconnaissance mission conducted by uav swarms based on distributed pso path planning algorithms. *IEEE access*, 7:105086–105099, 2019.

[111] Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9301–9308. IEEE, 2023.

[112] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[113] Shuhuan Wen, Zeteng Wen, Di Zhang, Hong Zhang, and Tao Wang. A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning. *Applied Soft Computing*, 110:107605, 2021.

[114] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

[115] Annie Wong, Thomas Bäck, Anna V Kononova, and Aske Plaat. Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review*, 56(6):5023–5056, 2023.

[116] KS Yakovlev, AA Andreychuk, Aleksei Aleksandrovich Skrynnik, and Aleksandr Igorevich Panov. Planning and learning in multi-agent path finding. In *Doklady Mathematics*, volume 106, pages S79–S84. Springer, 2022.

[117] Yang Yang, Li Juntao, and Peng Lingling. Multi-robot path planning based on a deep reinforcement learning dqn algorithm. *CAAI Transactions on Intelligence Technology*, 5(3):177–183, 2020.

[118] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[119] Jingjin Yu and Steven M LaValle. Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation*, pages 3612–3617. IEEE, 2013.

[120] Novak Zagradjanin, Dragan Pamucar, and Kosta Jovanovic. Cloud-based multi-robot path planning in complex and crowded environment with multi-criteria decision making using full consistency method. *Symmetry*, 11(10):1241, 2019.

[121] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[122] Yulun Zhang, Matthew C Fontaine, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Multi-robot coordination and layout design for automated warehousing. *arXiv preprint arXiv:2305.06436*, 2023.

[123] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

Appendix A

Additional Information

A.1 Preface

Chapter 2 of this thesis corresponds to work that has been published on *Artificial Intelligence Review* 57 (2), 41. The paper, entitled "Learning team-based navigation: a review of deep reinforcement learning techniques for multi-agent pathfinding," appears under the citation [11]. Chapter 4 and Chapter 5 of this thesis corresponds to the paper "The effectiveness of state representation model in multi-agent proximal policy optimization for multi-agent pathfinding" that has been accepted to the IROS 2024 conference.