Learning Cooperation for Partially Observable Multi-Agent Path Finding

by

Qiushi Lin

B.Eng., Southern University of Science and Technology, 2020

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Qiushi Lin**
Degree:	**Master of Science**
Thesis title:	**Learning Cooperation for Partially Observable Multi-Agent Path Finding**
Committee:	**Chair:** Angel Chang Assistant Professor, Computing Science
	Hang Ma Supervisor Assistant Professor, Computing Science
	Xue Bin Peng Committee Member Assistant Professor, Computing Science
	Oliver Schulte Examiner Professor, Computing Science

Abstract

In many real-world applications of multi-agent systems, decentralized agents have to cooperate to plan their collision-free paths under partial observation. Recently, many works have introduced multi-agent reinforcement learning (RL) to solve this partially observable variant of Multi-Agent Path Finding (MAPF) by learning homogeneous policies through centralized training and decentralized execution. However, complex multi-agent cooperation towards optimizing one or multiple objectives is hard to achieve by existing learning-based methods due to the curse of multiagency.

In this thesis, we aim to design algorithms that learn multi-agent cooperation for path planning towards various objectives. To tackle single-objective cooperation for partially observable MAPF, we propose Soft Actor-Critic with Heuristic-Based Attention (SACHA), a novel multi-agent actor-critic RL framework for the learned model to generalize among multiple instances. Moreover, we investigate the decentralized variant of another similar problem, Moving Agents in Formation (MAiF), that combines path planning with formation control. To learn bi-objective cooperation, we propose Mean Field Control with Envelop Q-learning (MFC-EQ), a scalable and adaptable learning framework for balancing two specific objectives among decentralized agents. We provide theoretical analysis to show the effectiveness of our methods and empirical evaluation to demonstrate that our methods can outperform baselines for numerous instances in various environments.

Keywords: Reinforcement Learning; Multi-Agent Systems; Path Planning

Acknowledgements

I would like to thank my supervisor Prof. Hang Ma, who has offered me great opportunities and freedom to explore the research world and has provided patient guidance and profound vision through my research journey at SFU. I would also like to thank all the committee members, Prof. Oliver Schulte and Prof. Xue Bin Peng, for their comments and suggestions on this thesis. I want to thank all past and current members of the SFU Robotics Research Group for their valuable help and insight. Last but not least, I would like to thank my parents, Wenhua Jiang and Jiachun Lin, for their unconditional love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent studies have demonstrated great success of Multi-Agent Path Finding (MAPF) [38] in various applications, such as warehouse and office robots [52, 47], autonomous aircraft-towing vehicles [29], and other multi-robot systems [10]. The canonical MAPF is an NP-hard combinatorial search problem [55, 2] that aims to plan collision-free paths for a set of agents from their start positions to their goal positions in a shared environment. The AI community has been developing many optimal and bounded-suboptimal MAPF planners, where a centralized planner is required to gather the complete environment information to plan joint paths for agents before execution. Nevertheless, these planners do not apply to decentralized agents with limited sensing capabilities and do not scale well to instances with large numbers of agents. Besides, generalizing these methods to solving multiple instances for optimizing more than one objective can be more challenging, though these scenarios are common in real-world applications.

In this thesis, we adopt learning-based approaches that are designed to learn cooperation for decentralized agents. We propose the learning frameworks based on the Multi-Agent Reinforcement Learning (MARL) paradigm. Although our works are framed in the context of MAPF, we believe the underlying insights could generalize to other partially observable tasks in homogeneous multi-agent systems.

1.1 Motivation

In many real-world applications of multi-agent systems, groups of agents are required to plan their collision-free paths from their start locations to their goal locations. The traditional search-based MAPF algorithms study this task as a combinatorial search problem, where all environment information is considered accessible, and they plan agents' entire paths before execution. Applying these algorithms to real-world instances raises several challenges. It is not necessarily the case that all environment information can be acquired and gathered to a centralized server, and even though that happens, solving this NP-hard search problem is computationally expensive. Also, once there occur small changes in the environment during execution, the system must halt and replan all agents' paths. It is more often the case that agents have mere capabilities of partial observation and limited communication and need to choose their sequential actions during execution. The partially observable environments limit agents' observation range to their surroundings, while the multi-agent communication can help reach the information outside one's observation but under limited capacity. Furthermore, it is sometimes required to optimize the cooperative paths over multiple objectives, and solving this type of task with search-based methods under the centralized paradigm is even more computationally intensive.

1.2 Contributions and Outlines

In this thesis, we consider a more practical setting, the partially observable MAPF, where decentralized agents can only observe their surrounding part of the environment and have to choose their actions based on their observation on the fly. Under this decentralized setting, we explore the use of the MARL algorithms to design learning frameworks for two specific tasks. The first task aims at one single objective of minimizing the sum of their path length. The second task is a bi-objective optimization problem where all agents are asked to reach their goals as quickly as possible while maintaining a particular formation as close as possible. We formulate these two tasks in Chapter 2.

While efforts have been devoted to designing learning-based methods for MAPF, the learned capability of multi-agent cooperation can be limited and hard to generalize among various environments. In Chapter 3, we propose a novel multi-agent actor-critic framework that incorporates the heuristic-based attention mechanisms for solving the partially observable MAPF. Our method encourages agents to take into account the surrounding agents' possible paths and select critical relevant information from them. Our learning framework also relies on an agent-centered critic network to judge each decentralized agent's action based only on local information, therefore the learned agent policy can be potentially generalized among different MAPF instances. Our results show that our method can outperform search-based algorithms within certain runtime limits and other learning-based methods.

To take the first step towards learning multi-objective cooperation for MAPF, in Chapter 4, we consider a bi-objective task, Moving Agents in Formation (MAiF), that integrates path planning with formation control. We combine the use of the mean field RL and the multi-objective RL to propose a scalable and adaptable learning framework. The mean field approximation is applied to tackle the complexity of multi-agent formation control, and the multi-objective RL enables decentralized agents to trade off two considered objectives under any linear preference. We empirically show that our method leads to better results than centralized baselines in most cases and can handle more challenging tasks with dynamic formations.

In this thesis, we propose two MARL frameworks to learn single-objective and bi-objective cooperation for large-scale instances within partially observable environments. The single-objective cooperation is devoted to generalization among various environments, while bi-objective cooperation focuses on adaptability toward different linear preferences of two considered objectives. There are more complicated forms of multi-objective cooperation, which will be left for future work. Our frameworks have been proven to provide great performance in MAPF but can be also of interest to other domains in homogeneous multi-agent systems.

Chapter 2

Background

In this chapter, we first formulate the standard Multi-Agent Path Finding (MAPF) and Moving Agents in Formation (MAiF) as combinatorial search problems. We then define the settings in which we study both problems.

2.1 Problem Definitions

We first define the standard MAPF problem as a single-objective search problem. Then we formulate MAiF, a bi-objective search problem proposed in [17]. Based on that, we introduce the partially observable environment and the homogeneous multi-agent systems in which we study these two problems.

2.1.1 Standard Multi-Agent Path Finding

In the standard formulation of MAPF, we are given a connected and undirected graph $G = (V, E)$ and a set of M agents, indexed by $i \in \{1, 2, \cdots, M\}$. Each agent has a unique start vertex $s_i \in V$ and a unique goal vertex $g_i \in V$. Time is discretized into time steps, $t = 0, 1, \cdots, \infty$. Between two consecutive time steps, each agent can either *move* to one of its adjacent vertices or *wait* at its current vertex. A path for agent i contains a sequence of vertices that lead agent i from s_i to g_i, where each vertex indicates the position of the agent for every time step. The *completion time* T_i of agent i is defined as the length of its path, and it is the earliest time when agent i has reached and terminally stayed at its goal vertex. Collisions between agents are not allowed. A vertex collision occurs when two agents occupy the same vertex v at the same time t. An edge collision occurs when two agents traverse the same edge (u, v) in opposite directions from t to $t + 1$. A MAPF solution is a set of collision-free paths for all agents. A commonly used objective function is the average (equivalently, sum) of the completion times of all agents.

2.1.2 Standard Moving Agents in Formation

In the standard formulation, an MAiF instance is defined on an undirected graph $G = (V, E)$ in a d-dimensional Cartesian system. Each location in V can be recognized by its coordinates $\boldsymbol{v} = (v_1, \dots, v_d) \in \mathbb{R}^d$. In this thesis, the subscripts represent agents' index numbers and the boldface font denotes multi-dimensional vectors. We also define $[M] = \{1, \dots, M\}$. We have a set of M agents $I = \{a^i | i \in [M]\}$. Each agent has a unique start location $\boldsymbol{s}^i \in V$ and goal location $\boldsymbol{g}^i \in V$. The time is discretized, and between two consecutive time steps, each agent can choose to wait at the current location or move from $\boldsymbol{v}$ to $\boldsymbol{v'}$ provided that $(\boldsymbol{v}, \boldsymbol{v'}) \in E$. We also consider two types of collision between agents: a vertex collision $\langle a^i, a^j, \boldsymbol{v}, t\rangle$ means agent a^i and a^j occupy the same location at the same time step t, and an edge collision $\langle a^i, a^j, \boldsymbol{u}, \boldsymbol{v}, t\rangle$ happens when a^i travels from $\boldsymbol{u}$ to $\boldsymbol{v}$ while a^j travels backward.

The MAiF problem aims to find a set of M collision-free paths $\Pi = \{\Pi^i | i \in [M]\}$ as a solution, where $\Pi^i = (p^i_0, \dots, p^i_{T^i})$ represents agent i's trajectory. Every solution will be evaluated by two objectives, makespan and formation deviation. The makespan can be defined as $T = \max_{1 \le i \le M} T^i$, that is, the longest length among all paths. The *formation* at time t can be represented as an M-tuple, $\ell(t) = \langle \boldsymbol{p}^1(t), \dots, \boldsymbol{p}^M(t)\rangle$. The desired formation is the combination of all agents' goal locations, $\ell_{\boldsymbol{g}} = \langle \boldsymbol{g}^1, \dots, \boldsymbol{g}^M\rangle$. Following the definition in [17], the *formation distance* between any two formation $\ell = \langle \boldsymbol{u}^1, \dots, \boldsymbol{u}^M\rangle$ and $\ell' = \langle \boldsymbol{v}^1, \dots, \boldsymbol{v}^M\rangle$ indicates the least effort required to transform from ℓ to ℓ', defined as:

$$\begin{aligned}\mathscr{F}(\ell, \ell') &:= \min_{\boldsymbol{\Delta}} \sum_{i=1}^{M} \|\boldsymbol{u}^i - (\boldsymbol{v}^i + \boldsymbol{\Delta})\|_1 \\ &= \sum_{i=1}^{M} \underbrace{\sum_{j=1}^{d} |(\boldsymbol{u}^i_j - \boldsymbol{v}^i_j) - \boldsymbol{\Delta}_j|}_{\overset{\text{def}}{=} \mathscr{F}^i(\ell, \ell')}, \end{aligned} \tag{2.1}$$

where j indexes the dimension for each vector and $\Delta_j = \mathit{median}(\left\{\boldsymbol{u}^i_j - \boldsymbol{v}^i_j\right\}_{i \in [M]})$ is the median for the j-th dimension. The $\mathscr{F}^i(\ell, \ell')$ denotes the subpart only dedicated to the agent a^i. We consider the average formation deviation per agent across all time steps, unlike the total formation deviation in [17], and it can be defined as $\mathscr{F}_{avg} = \frac{1}{M} \sum_{t=0}^{T} \mathscr{F}(t)$, where $\mathscr{F}(t) = \mathscr{F}(\ell(t), \ell_{\boldsymbol{g}})$. We also consider a mix of these two objectives:

$$\texttt{MIX}(\lambda) = \lambda T + (1 - \lambda)\mathscr{F}_{avg},$$

which is the linear combination of these two objectives. There could be non-linear combinations of different rewards, but, in this work, we only consider the linear cases, as it has been considered by many related works in multi-objective reinforcement learning and others in multi-task reinforcement learning (e.g., [3]).

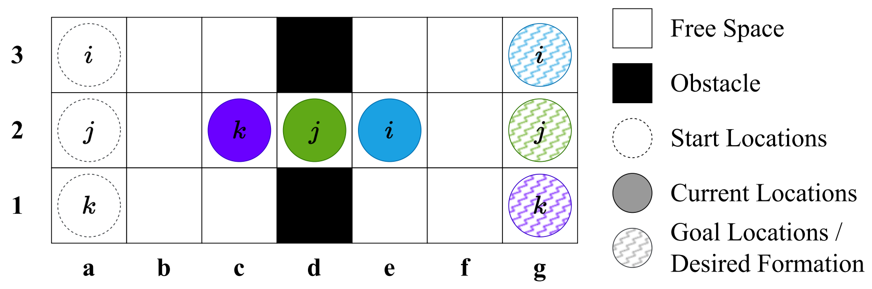

Figure 2.1: Example of moving agents in formation.

Example A simple MAiF example is demonstrated in Fig. 2.1. The start formation is $\langle a3, a2, a1\rangle$ and the goal/desired formation is $\langle g3, g2, g1\rangle$. The group of agents cannot go through the d column while keeping the formation intact, so they have to change the formation. At the current time step t, the median position is $d2$ and the formation deviation is $\mathscr{F}(t) = \mathscr{F}^i(t) + \mathscr{F}^j(t) + \mathscr{F}^k(t) = 2 + 0 + 2 = 4$.

2.1.3 Partially Observable Environments

We consider a more practical problem setting where, instead of assuming the full knowledge of the environment, each agent can only have a partial observation of its surroundings. We formulate decentralized MAiF as a decentralized partially observable Markov Decision Process (Dec-POMDP) [20]. A Dec-POMDP can be represented as a 7-tuple $\langle \mathcal{S}, \boldsymbol{A}, P_S, \boldsymbol{O}, P_O, R, \gamma\rangle$, where $\mathcal{S}$ is the global state space. $\boldsymbol{A} = \prod_{i=1}^{M} A^i$ and $\boldsymbol{O} = \prod_{i=1}^{M} S^i$, where A^i and S^i are agent i's action and observation space. $P_S : \boldsymbol{A} \times \mathcal{S} \to \mathcal{S}$ describes the state-transition function, and $P_O : \boldsymbol{A} \times \mathcal{S} \to \boldsymbol{O}$ is the observation-transition function. R is the reward function with the discount factor γ $(0 \leq \gamma < 1)$.

In partially observable environments, to get close to the traditional setups of path planning problems, we assume that the observation-transition and the state-transition functions are deterministic. Following the settings in existing learning-based MAPF methods, we formalize this problem on 2-dimensional 4-neighbor grids, even though our method can also easily generalize to other environments.

2.1.4 Homogeneous Multi-Agent Systems

In this thesis, we focus on designing learning algorithms for homogeneous multi-agent systems, meaning that agents share the same policy network and only act differently due to their different local observations. Since all decentralized agents are equal and cooperative towards the same objectives, learning a set of different agent policies can be approximated by learning one singular homogeneous policy, given that we train the policy over sufficient instances in a great variety of environments. The assumption enables our learned policy to generalize among various instances in different environments.

2.2 Cooperative Multi-Agent Reinforcement Learning

Our problems are structured in the paradigm of multi-agent reinforcement learning. Let $\boldsymbol{\pi} = [\pi_1, \dots, \pi_M]$ denote the joint policy for all agents, where $\pi^i : S^i \times A^i \to \mathbb{R}$ is the policy for agent i. At each control step t, each agent i takes one action a^i from π^i on the joint state s_t. The environment will respond with the next state s_{t+1} and a set of individual rewards $[r_t^1(s^1, a^1), \dots, r_t^M(s^M, a^M)]$.

The cooperative MARL aims to maximize the global reward, which is the sum of all individual rewards, $r_t(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^M r_t^i(s^i, a^i)$. The long-term cumulative reward can be denoted as the discounted sum of all global rewards:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\{s_t, a_t\} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{\infty} \gamma^t r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) \Big].$$

We also consider the value function and the state-action value function. The joint value function $V^{\boldsymbol{\pi}}(\boldsymbol{s})$ represents the estimated cumulative reward by following a policy $\boldsymbol{\pi}$ starting at a given global state $\boldsymbol{s}$:

$$V^{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}_{\{s_t, a_t\} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{T} \gamma^t r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_0 = \boldsymbol{s} \Big].$$

The state-action value function, referred as Q-function, represents the estimated cumulative reward by following a policy $\boldsymbol{\pi}$ starting at a given state $\boldsymbol{s}$ with a particular action $\boldsymbol{a}$:

$$Q^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\{s_t, a_t\} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{T} \gamma^t r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a} \Big].$$

The relationship of these two functions can be stated as:

$$V^{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{\pi}(s, \cdot)} \Big[Q^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) \Big].$$

We also define the advantage function as:

$$A^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) = Q^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) - V^{\boldsymbol{\pi}}(\boldsymbol{s}).$$

These functions can also be decomposed to every individual agent and learned locally through various approaches.

Chapter 3

Single-Objective Cooperation

3.1 Introduction

Although MAPF is NP-hard to solve optimally [55, 2], the AI community has developed many optimal and bounded-suboptimal MAPF planners for fully observable environments, where a centralized planner has complete information of the environment to plan joint paths for agents. These planners do not apply to agents with limited sensing capabilities and do not scale well to large numbers of agents, as the complexity of coordinating the joint paths of agents grows exponentially with the number of agents in the systems. Learning-based methods with centralized training and decentralized execution have been proposed to develop scalable and generalizable learning-based MAPF methods for the partially observable setting. In this setting, each agent receives a partial observation of its surroundings. Learning-based MAPF methods aim to train a decentralized homogeneous policy that each agent will follow based on its local observation during execution. This policy can be distributed to any number of agents in any environment, as the dimension of the single-agent observation space depends only on the FOV size in the partially observable setting. However, the non-stationarity of environments from the perspective of any single agent poses a significant challenge for learning-based MAPF methods. The transitions of the global state are affected by the individual actions of other agents towards their local interests. Moreover, goal-oriented reinforcement learning with single-agent rewards makes the training process unstable and time-consuming, further incentivizing each agent to be selfish and prioritize its goal over collaborating with others. This could hinder coordination and teamwork among agents, negatively affecting overall performance.

To address the challenges posed by solving MAPF in the partially observable setting, we propose **S**OFT **A**CTOR-**C**RITIC WITH **H**EURISTIC-BASED **A**TTENTION (SACHA), a novel approach for partially observable MAPF that leverages heuristic guidance through attention mechanisms to learn cooperation. SACHA builds upon the multi-agent actor-critic framework and, along with its communication-based alternative, SACHA(C), aims to learn a decentralized homogeneous policy that can be generalized to any number of agents in any

Table 3.1: Comparison of learning-based methods for partially observable MAPF

Method	**Learning Framework**	**Communication Moudule**	**Single-Agent Guidance**	**Cooperative Guidance**
PRIMAL [34]	A3C (RL) and Behavior Cloning (IL)	Inapplicable	Goal Direction	Goal Directions of Neighbouring Agents
DHC [25]	IQL	Required	Shortest Path Distance	–
DCC [26]	IQL	Required	Shortest Path Distance	–
SACHA (Ours)	Mutli-Agent Soft Actor-Critic	Optional	Shortest Path Distance	Shortest Path Distances of Neighbouring Agents

arbitrary partially observable MAPF environment. To achieve this, we first allow each agent to access the goal-oriented heuristic guidance of multiple agents in the form of the shortest path distances to each of the goals, which can be computed efficiently before execution. We then employ a self-attention module in the policy network for each agent to locally select relevant information from the guidance and take actions towards better cooperation among agents.

We expect SACHA to make a significant algorithmic impact not only on MAPF solving but also on other similar multi-agent tasks in partially observable settings because its learning process of the homogeneous policy is also guided by a homogeneous critic for more stable learning and faster convergence. Unlike existing multi-agent actor-critic methods with one fully centralized critic or multiple decentralized critics, SACHA introduces a novel agent-centered critic network that uses an attention mechanism to approximate each agent's Q-function and performs credit assignment only based on the information within each partial observation. The input dimension of this critic is determined by the number of agents that each agent's Q-function should be based on, which is implicitly limited by the partial observation range (e.g., FOV size in MAPF). This partially centralized critic ensures that the Q-function is not biased towards any specific problem instance, resulting in a well-trained policy network that can generalize well to different numbers of agents and environments.

We experimentally compare SACHA and SACHA(C) with state-of-the-art learning-based and search-based MAPF methods over several MAPF benchmarks. Our results show that both versions of SACHA result in higher success rates and better solution quality than other methods in almost all test cases. The results thus indicate that our methods allow for better cooperation among agents than the other methods with and even without communication.

3.2 Related Work

We now survey the related work on learning-based MAPF methods and multi-agent reinforcement learning methods.

3.2.1 Learning-Based MAPF Solvers

Recent learning-based MAPF methods [34, 21, 25, 26] have been proposed to solve MAPF in a partially observable setting. These methods aim to learn a decentralized policy that can be generalized to different MAPF instances. While centralized MAPF planners require full observation of the environment and must plan paths from scratch for each instance, the well-trained model can be applied to MAPF instances with any number of agents and environment size, without increasing the time complexity.

The most straightforward approach for tackling partial observability is to treat other agents as part of the environment and let each agent learn its policy independently, as in Independent Q-Learning (IQL) [44]. However, this approach results in non-cooperative behavior among the agents, and its training is not guaranteed to converge due to interference between the policies of different agents. State-of-the-art MAPF methods [25, 26] enhance IQL with a communication mechanism to promote cooperation between agents. Other MAPF methods [34, 21] use the actor-critic framework, with guidance from an expert demonstration. PRIMAL [34] combines on-policy asynchronous advantage actor-critic (A3C) [28] with behavior cloning from an expert demonstration generated by a centralized MAPF planner [48]. However, the centralized MAPF planner requires solving numerous MAPF instances, which slows down the training process. DHC [25] and DCC [26] have shown that using single-agent shortest path distances as heuristic guidance for goal-oriented learning of each agent is more effective than following a specific reference path in a multi-agent cooperative setting.

In Section 3.6, we compare SACHA against PRIMAL, DHC, and DCC experimentally. Table 3.1 summarizes the comparison of properties of these methods, showing that SACHA improves over them by adopting a more stable training scheme, utilizing better heuristic guidance through more complex model design, and allowing for applicability to both communicating and non-communicating scenarios.

3.2.2 Cooperative Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is a well-established framework for coordinating multiple agents in a shared environment. A rich literature [43, 11, 31] on cooperative MARL has been dedicated to coordinating agents that work towards a common objective and take actions that benefit all agents as a whole. To deal with the nonstationarity of the environment, most existing actor-critic methods use one or more fully centralized critics that observe the entire environment. For example, Multi-Agent Deep Deterministic Policy Gradient [22] trains each actor with only its local observation using the DDPG algorithm [19], while its corresponding fully centralized critic can access the observations and actions of all agents. Instead of using multiple fully centralized critics, Counterfactual Multi-Agent Policy Gradient [8] uses only one fully centralized critic that learns to assign credit to agents and estimate Q-functions for all agents based on a counterfactual baseline that marginalizes out the action of each agent. However, it becomes increasingly difficult to perform such credit assignments for cases with large numbers of agents. Therefore, Multiple Actor Attention-Critic[15] deploys an attention mechanism for the fully centralized critic to selectively pay attention to relevant information from all agents. SACHA also uses a similar attention mechanism but differs from existing actor-critic methods by using a novel homogeneous agent-centered critic that only takes the local information from each agent as input for generalizability to different MAPF instances instead of a fully centralized critic specific to only one MAPF instance.

3.3 Environment and Model Design

In MAPF, the observation and state-transition functions are deterministic, where each agent has full control of its next position and observation by taking one of the move actions or the wait action. To facilitate proper comparison with existing learning-based MAPF methods, we formalize the MAPF problem on two-dimensional grid maps with four neighbors, even though our method can also be generalized to other MAPF problems. The partial observability limits each agent's perception to its FOV, defined as a $\mathcal{L} \times \mathcal{L}$ square area centered on the agent. Agents take their actions based on their local observation and the history from the beginning to the current time step.

One of the key challenges for decentralized planners with limited access to global information is the occurrence of deadlocks and livelocks. Similar to time-windowed MAPF planners [18], these issues arise due to the limited planning horizon of agents, either in time or space, that prevents them from reaching their goals. For instance, consider Fig. 3.1, where the red agent in $a5$ and the green agent in $a6$ are heading towards their respective goals $a6$ and $a5$. An optimal solution may require the red agent to move north and terminally stay at its goal while the green agent moves north and takes a detour since its direct path is blocked by the red agent. However, after a few moves, the green agent will no longer observe the blocking red agent and end up wiggling between $a9$ and $a10$ indefinitely. Symmetrically, if the green agent moves south and terminally stays at its goal, the red agent will eventually wiggle between $a1$ and $a2$.

Existing learning-based MAPF methods rely on two extra assumptions to alleviate the above issues: First, each agent has full visibility of the map (which is consistent with both standard MAPF and time-windowed MAPF), even though it does not know the global state. Second, two agents are allowed to communicate when they are within each other's FOV. In this work, SACHA and SACHA(C) utilize the same assumptions. Both methods give each agent access to the shortest path distances to its goal. SACHA(C) enables inter-agent communication, while SACHA only requires each agent to identify other agents in its FOV.

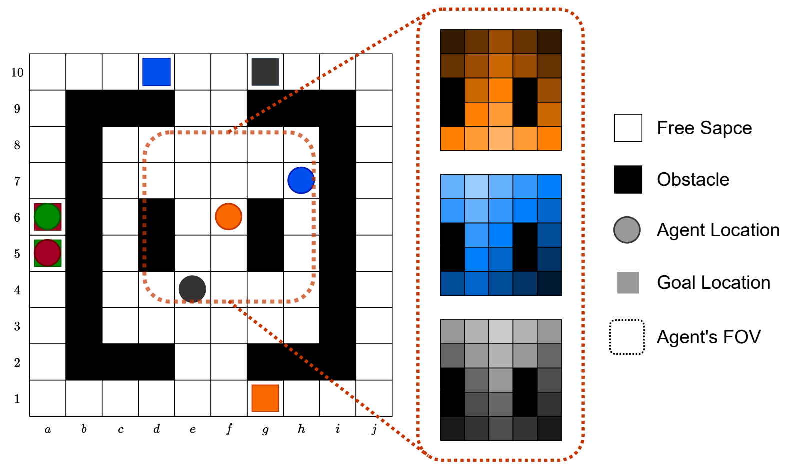

Figure 3.1: Illustration of a partially observable MAPF instance and the multi-agent heuristic maps for the orange agent. Each agent's current (circle) and goal (square) positions have the same color. The orange agent's 5×5 FOV contains 3 agents, including itself. Therefore, the policy network of the orange agent will utilize 3 corresponding heuristic maps within the same 5×5 area. A darker shade of color represents a greater distance to the goal.

3.3.1 Multi-Agent Shortest Path Distance Heuristic Maps

Empirical studies [26] have shown that shortest path distances from all vertices to each agent's goal vertex can greatly benefit goal-oriented learning for the agent in partially observable environments. SACHA utilizes multi-agent shortest path distances to guide not only the achievement of single-agent goals but also better cooperation between agents. Specifically, a backward uniform-cost search is run from each agent's goal vertex to all vertices in the given graph to generate the shortest path heuristic maps for the agent. The heuristic maps for all agents can be calculated offline once the graph and all goal vertices are given. They remain unchanged for the same MAPF instance during training and can be efficiently generated in advance for a new MAPF instance during execution. The time complexity for calculating the distances for M agents on graph $G = (V, E)$ is $\mathcal{O}(M|E|\log|V|)$. Many search-based and some recent learning-based MAPF methods [35, 25, 26] also use heuristic maps of shortest path distance, but only as single-agent guidance. SACHA gives each agent access to not only its heuristic map but also the heuristic maps of other agents within its FOV, which enables better cooperation. Fig. 3.1 visualizes the heuristic maps that the orange agent has access to at the current time step. Since there are three agents within its FOV, three corresponding heuristic maps are input to the agent's policy network which we will describe later.

3.3.2 Reward Design with Heuristic Maps

We design the reward function for each agent based on its heuristic map. We start with the individual reward function of DHC [25]:

$$r_i(s,a) = \begin{cases} -0.075 & \text{move (up / down / left / right) and wait (away goal)} \\ 0 & \text{wait (on goal)} \\ -0.5 & \text{collision (with obstacles or agents)} \\ 3 & \text{reach goal} \end{cases}.$$

It follows the intuition that an agent is punished slightly for each time step before arriving at the goal, thereby encouraging it to reach its goal as quickly as possible. To improve the success rate of solving the global MAPF task, each agent is punished to a greater extent each time it collides with another agent or an obstacle. The agent receives a positive reward only when it arrives at its goal. Most existing learning-based MAPF methods follow the same design idea for their reward functions. However, the goal-conditioned reward in this reward design makes the training unstable and difficult to converge, especially for long-horizon tasks such as MAPF. Also, since an agent that stays further away from its goal generally has a larger potential to collide with other agents, the move actions for it should not be rewarded the same as for ones that are closer to their goals. Therefore, motivated by Heuristic-Guided Reinforcement Learning (HuRL) [7], we reshape the reward function for each agent with an additional heuristic term. Assume we have a transition tuple, (s, a, r, s'), we reshape the reward as:

$$\tilde{r}_i(s,a) = r_i(s,a) + (1-\lambda)\gamma h_i(s'), \tag{3.1}$$

where $h_i(s')$ is the negated normalized shortest path distance of the global state s' from the heuristic map of agent i. This heuristic term represents a priori guess of the desired long-term return of an agent from state s' and thus serves as a horizon-based regularization. HuRL has been proved both theoretically and empirically to be able to accelerate the learning process significantly by intrinsically reshaping the reward of every position for each agent. We set λ to 0.1 in the experiments.

3.3.3 Model Design with Attention Mechanisms

We propose a novel model architecture based on the multi-agent actor-critic framework. Our model aims to achieve generalization across different instances by restricting the actor and the critic to operate only within the observation of each agent. At each time step t, we define an undirected observation graph $G_t = (V, E_t)$, where V is the set of all agents and each edge in E_t indicates that the corresponding agents can observe each other. The time-varying graph G_t captures the dynamic correlation of agents in partially observable

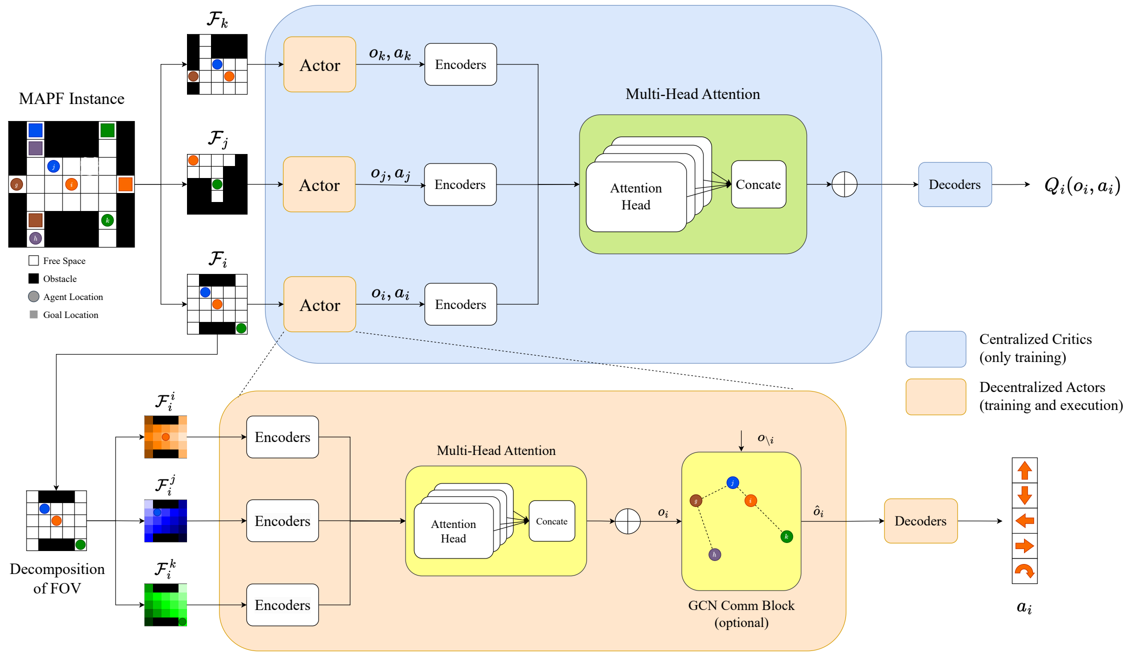

Figure 3.2: Model design of SACHA and SACHA(C). The framework consists of the actors (red) and the critics (blue). The bottom part demonstrates agent i's policy network. Within its 5×5 FOV, there are three agents inside and hence three heuristic maps are fed into its network. After going through a set of encoders, three output features will then be processed by the attention block, picking out the relevant information required for agent i to take its action. The upper part shows how the critic assigns the credit for agent i based on the local observations and the corresponding actions from agents in its subgroup via a similar attention mechanism.

environments. We denote the subgroup centered on agent i as $\{i \cup \mathcal{N}_t(i)\}$, where $\mathcal{N}_t(i)$ is the set of the nearest $K-1$ neighbors of node i inside its FOV. The observation of each agent i consists of a set of K feature maps $\mathcal{F}_i = \{\mathcal{F}_i^j\}_{j \in i \cup \mathcal{N}_t(i)}$ where $\mathcal{F}_i^j \in \mathbb{R}^{\mathcal{L} \times \mathcal{L} \times 3}$. Each feature map in the set corresponds to an agent in the subgroup of agent i and contains three channels: (1) a binary matrix that identifies the obstacles and the free space, (2) a binary matrix that marks the positions of agents, and (3) a heuristic channel that shows the normalized shortest path distances for each empty cell. K is set to 3 in our experiments.

We present the learning framework of SACHA in Figure 3.2. Given the observation features, the policy network starts with the observation encoders with shared parameters. The encoders consist of several convolutional layers followed by a GRU [7] memory unit. The output set of K encoding is input into the Multi-Head Attention (MHA) [46] module that learns the interaction between agent i and its subgroup members by selectively attending to relevant information. The MHA module outputs a set of features the sum of which is used for the observation representation, denoted as o_i. It then will be passed to decoders to generate the corresponding action vector $a_i \in \mathbb{R}^5$. Each element in a_i represents the probability of choosing one of the five discrete actions $\{up, down, left, right, stay\}$.

We propose a novel agent-centered critic, that evaluates each agent's action individually based on its local observation and information about its subgroup members. Unlike previous methods that use a centralized critic with global information, our critic leverages the attention mechanism to dynamically assign credit to each agent. We first pass the policy network's output through a linear function and then apply a multi-head attention module. The sum of the concatenated output vectors is then forwarded through the decoders to obtain the final Q-value, which is used to update the policy networks via the policy gradient method. Since our agent-centered critic only requires the local information of the central agent, it is not dependent on any specific environment information, and the learned policy network can thus generalize better.

3.3.4 Optional Communication Module

Furthermore, we propose a communication-based variant of our method, named SACHA(C). To encourage better cooperation, our method should be able to take advantage of communication when it is allowed. We add an optional communication module after the multi-head attention block, as shown in Fig. 3.2. We first gather all observation representation $\{o_i\}_{i=1}^{M}$ as the initialization ,$H^{(0)}$, and then feed it into a multi-layer Graph Convolution Network [16] (GCN). Recall that M is the number of agents. Let A_t be the adjacency matrix of G_t and $\tilde{A}_t = A_t + I_M$. Define $\tilde{D}_t$ as a diagonal matrix where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The output of the $(l+1)$-th layer would be:

$$H^{(l+1)} = \sigma(\tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}} H^{(l)} W^{(l)}), \tag{3.2}$$

where $\sigma(\cdot)$ is the sigmoid function and W is a layer-specific trainable weight matrix. After l layers of GCN, we can decompose $H^{(l+1)}$ to M corresponding vectors, $\{\hat{o}_i\}_{i=1}^{M}$, which will eventually be decoded by each network to their corresponding action vectors as usual. We choose a two-layer GCN in the SACHA(C). The communication module is optional, but it can make agents reach information outside their local observation and hence achieve better cooperation.

3.4 Learning Framework

SACHA updates the agent's policy network π parameterized by θ and the critic network parameterized by ψ simultaneously through the soft actor-critic framework. We let $\bar{\theta}$ and $\bar{\psi}$ denote the moving average of θ and ψ (target parameters of the actor and the critic network), respectively. We first define the action-value temporal difference (TD) error for any experience, $e = (s, a, \tilde{r}, s')$, from the replay buffer D:

$$\delta_i = Q_i^{\psi}(o_i, a_i) - \tilde{r}_i - \gamma \mathbb{E}_{a_i' \sim \pi_{\bar{\theta}}(o_i')}[Q_i^{\bar{\psi}}(o_i', a_i') - \alpha \log(\pi_{\bar{\theta}}(a_i'|o_i')))] \tag{3.3}$$

where α is the temperature parameter that decides the weight of the entropy term in the soft actor-critic framework [12]. SACHA runs the critic network through every agent-centered subgroup and updates it by minimizing the mean square error loss function:

$$L_Q(\psi) = \mathbb{E}_{e \sim D} \sum_{i=1}^{M(e)} \frac{\delta_i^2}{M(e)}, \tag{3.4}$$

where $M(e)$ is the number of agents in e. On the other hand, the actors update their underlying policy networks by the policy gradient via the Q-values from the critic network:

$$\nabla_{\theta_i} J(\theta) = \mathbb{E}_{o_i \sim D, a_i \sim \pi_{\theta_i}(o_i)}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i))(Q_i^{\psi}(o_i, a_i) - b(o_i, a_{\setminus i}) - \alpha \log(\pi_{\theta_i}(a_i|o_i)))], \tag{3.5}$$

where $\nabla_\theta \log(\pi_\theta(a_i|o_i))$ is the score function and $Q_i^{\psi}(o_i, a_i) - b(o_i, a_{\setminus i})$ is the multi-agent advantage function. Inspired by COMA [8], SACHA adopts the counterfactual baseline in a discrete action space as follows, which marginalizes out the specific action of agent i:

$$b(o, a_{\setminus i}) = \sum_{a_i' \in \mathcal{A}_i} \pi(a_i'|o_i) Q_i^{\psi}(o_i, (a_i', a_{\setminus i})), \tag{3.6}$$

where $a_{\setminus i} \in \mathcal{A}_{\setminus i} = \prod_{j \neq i} \mathcal{A}_j$ is the combination of actions from all agents except for i and $\mathcal{A}_i$ is each agent's action space. This baseline specifically compares an action to other actions of agent i by fixing the actions of all other agents and invoking the critic network for $|\mathcal{A}_i|$ times. In each instance, we collect these M updated policies and aggregate them into the new policy by averaging all the locally updated policies: $\theta^{(t+1)} = \sum_{i=1}^{M} \frac{\theta_i^{(t)}}{M}$, where $\theta_i^{(t)} = \theta^{(t)} - \nabla_{\theta_i} J(\theta^{(t)})$. The policy and the critic network are updated together iteratively to reach fast and stable convergence.

3.5 Analysis

In this section, we analyze the effectiveness of the policy gradient in our multi-agent actor-critic framework. Most of the existing learning-based MAPF methods use IQL, in which each agent treats others as part of the environment and updates the global policy θ as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim D, a \sim \pi_\theta}[\nabla_\theta \log(\pi_\theta(a|s)) Q(s, a)], \tag{3.7}$$

where s and a denote the joint state and joint action, respectively. Now we show that this gradient is equivalent to Eq. (3.5). We omit the entropy term here, but the proof can be easily extended to include it.

Since each agent acts on its policy independently, we have $\pi_\theta(a|s) = \prod_i \pi_{\theta_i}(a_i|s)$. We represent the stationary distribution induced by π_θ as $d_\theta(s)$, meaning the probability of being in the state s by following π_θ. Following the proof in [42], we get:

$$\begin{aligned}
\nabla_{\theta_i} J(\theta) &= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\nabla_{\theta_i}\Big[\sum_{j=1}^{M}\log(\pi_{\theta_j}(a_j|s))\Big]Q(s,a)\\
&= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\nabla_{\theta_i}\log(\pi_{\theta_i}(a_i|s))Q(s,a)\\
&= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \Big[\prod_{j\neq i}\pi_{\theta_j}(a_j|s)\Big]\nabla_{\theta_i}\pi_{\theta_i}(a_i|s)Q(s,a).
\end{aligned}$$

To further the proof, we consider the following equation:

$$\begin{aligned}
&\sum_{a\in\mathcal{A}} \Big[\prod_{j\neq i}\pi_{\theta_j}(a_j|s)\Big]\nabla_{\theta_i}\pi_{\theta_i}(a_i|s)F(s,a_{\setminus i})\\
=&\sum_{a_{\setminus i}\in\mathcal{A}_{\setminus i}} \Big[\prod_{j\neq i}\pi_{\theta_j}(a_j|s)\Big]F(s,a_{\setminus i})\Big[\nabla_{\theta_i}\underbrace{\sum_{a_i\in\mathcal{A}_i}\pi_{\theta_i}(a_i|s)}_{=1}\Big]\\
=&\,0.
\end{aligned}$$

This will stay true as long as $F(s,a_{\setminus i})$ is a function independent of a_i. Let $F(s,a_{\setminus i}) = -Q(s,a_{\setminus i}) - b(s,a_{\setminus i})$ and combine it with the equation above:

$$\begin{aligned}
\nabla_{\theta_i} J(\theta) &= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \Big[\prod_{j\neq i}\pi_{\theta_j}(a_j|s)\Big]\nabla_{\theta_i}\pi_{\theta_i}(a_i|s)\cdot[Q(s,a)-Q(s,a_{\setminus i})-b(s,a_{\setminus i})]\\
&= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \Big[\prod_{j\neq i}\pi_{\theta_j}(a_j|s)\Big]\nabla_{\theta_i}\pi_{\theta_i}(a_i|s)\cdot[Q_i(s,a_i)-b(s,a_{\setminus i})]\\
&= \sum_{s\in D} d_\theta(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\nabla_{\theta_i}\log(\pi_{\theta_i}(a_i|s))\cdot[Q_i(s,a_i)-b(s,a_{\setminus i})]
\end{aligned}$$

Here, we prove that the policy gradient with respect to each θ_i can be obtained locally using the corresponding score function, $\nabla_{\theta_i}\log(\pi_{\theta_i}(a_i|s))$. By averaging $\theta_i^{(t)}$ from all agents updated by $Q_i(o_i,a_i)-b(o_i,a_{\setminus i})$, we obtain the same effect as updating the global θ based on $Q(s,a)$. Therefore, our method is as effective as IQL, but with a faster convergence rate due to parallel updates among all agents. Moreover, our framework for learning a homogeneous policy from local observation in the multi-agent learning framework is not restricted to MAPF and can potentially be applied to other MARL tasks, especially ones in partially observable settings.

3.6 Empirical Evaluation

In this section, we implement MFC-EQ and experimentally evaluate it with other methods on a server equipped with Intel 2.3GHz 16-Core CPUs and NVIDIA A40 GPUs.

3.6.1 Environment Setups

Training Environments: As mentioned above, our model is trained using the multi-agent actor-critic learning framework. Not only does each agent's policy network share parameters but also the critic networks applied to each subgroup of agents are homogeneous. We train our model over random grid maps of different sizes with randomly generated obstacles. The obstacle density is sampled from a triangular distribution between 0 and 0.5 with its peak value at 0.33. To fairly compare with other decentralized MAPF planners, our agent's policy network exclusively has 9×9 square-shaped FOV, the same as DHC and DCC, regardless of the environment size. Inspired by the curriculum learning [4], we design a training pipeline that starts with only 2 agents on 10×10 grid maps and gradually increases the number of agents and the size of the map once the success rate reaches a certain threshold. More and more complicated tasks are constantly added to the training pool until the map size exceeds 100×100 or the number of agents exceeds 72.

Testing Environments: We test our methods over a variety of maps all from the standard benchmark [38]. We first select two random maps (32×32 and 64×64) with uniformly distributed obstacles. Besides, we also use two game maps, **den312d** (65×81) and **warehouse** (161×63). The start-goal pairs of agent locations are randomly generated with the guaranteed existence of solutions. The number of agents is chosen from $\{4, 8, 16, 32, 64\}$ respectively. The maximum time step is 256 for **random32**, **random64**, and **den312d**, and 512 for **warehouse**. For cases that cannot be solved successfully within the time horizon or the runtime limit, we count each agent's step as the maximum time step.

3.6.2 Baselines

Learning-Based Methods: We compare our methods with several state-of-the-art decentralized learning-based MAPF methods summarized in Table 3.1. PRIMAL uses expert demonstration from centralized MAPF planners to train its model. The expert demonstration has a positive effect on speeding up the training process but is very time-consuming and requires global information about the specific environment, which limits its generalization to unseen instances. DHC adopts IQL along with single-agent heuristic guidance and broadcast communication mechanism. The resulting model performed better than PRIMAL without any experts. DCC improves on DHC by learning selective communication with a decision causal unit, which can filter out redundant messages and focus on relevant information. This also reduces the communication frequency significantly.

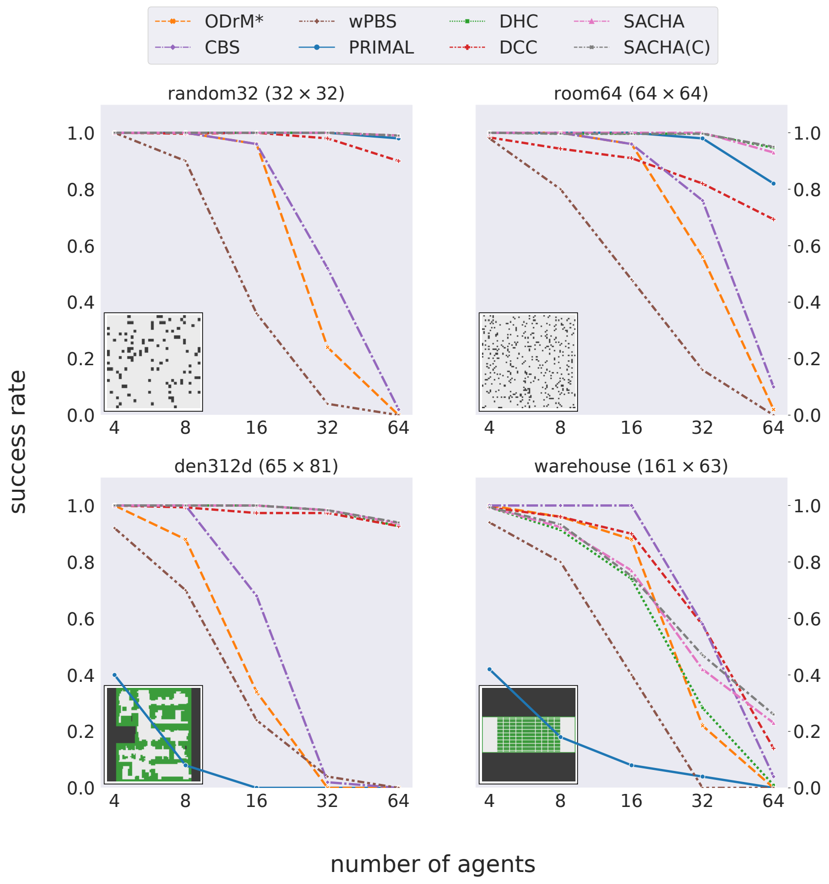

Figure 3.3: Success rates for different learning-based MAPF methods.

Search-Based Methods: Furthermore, we also compare our method with the centralized planners. We use two optimal but computationally expensive search-based methods for comparison: Conflict-Based Search (CBS) [35] and ODrM* [49]. CBS performs a best-first search that expands the constraint tree node by adding constraints to each agent involved in every conflict, while ODrM* is a suboptimal planner based on M* that applies Operator Decomposition (OD) to split agents into independent conflict sets and thus reduce the complexity of joint planning. We also use Priority-Based Search (PBS) [23] that searches for certain agent orders that can be used in the prioritized planning, which makes this solver incomplete but efficient. To simulate centralized planning in the same partially observable setting, we use windowed PBS (wPBS) [18], which only avoids conflicts within a bounded time horizon. We set the time window length of wPBS to be equal to the caliber of the FOV to simulate the partially observable environments. We set the runtime limit of CBS and wPBS to 120 seconds and ODrM* to 20 seconds.

3.6.3 Main Results

We evaluate the performance of the MAPF methods based on two widely-used metrics, success rate and average step per agent. Success rate measures the ability to solve the given instances within the runtime limit, whereas the average step per agent measures the quality of the solutions over a given set of instances. We test our approach along with multiple baselines in around 300 MAPF instances for each map with different numbers of agents.

Fig. 3.3 shows the success rate of our methods compared with all other baselines over four different MAPF maps. Within the time limit, all decentralized planners have a remarkable advantage in success rate. Even by including the precomputing time of shortest path heuristics, decentralized planners find solutions much faster than centralized ones. Among decentralized planners, PRIMAL tends to result in solutions with the worst quality, especially in those two game maps, which indicates that learning from the expert data cannot be easily generalized to instances with different numbers of agents and on maps with different structures. DHC and DCC have their advantages over PRIMAL by allowing the shortest path heuristics and the communication mechanism, although our methods can both outperform them in most cases. The advantages are more obvious over maps with higher obstacle density and more agents where more cooperation is demanded.

Table 3.2 reports the average step required to finish goals from each agent in multiple different instances. If planners exceed the runtime limit in some cases, we consider them failures and count them as spending the maximum time horizon. The stroke-out data with maximum time steps indicates zero success. When compared to search-based planners with relatively small numbers of agents, as expected, all learning-based methods cannot provide comparable results. However, as the number of agents grows, the search-based methods are significantly more time-consuming than learning-based methods and the success rate would drop rapidly due to the runtime limit. The two communication-based methods DHC and DCC have greater solution quality over PRIMAL in all cases. However, SACHA and SACHA(C) outperform them by a decent margin in most instances, with and without the communication block, which demonstrates their advantages over other learning-based methods. It is worth mentioning that, generally, SACHA(C) has better performance than SACHA in instances with larger numbers of agents where communication can be rather helpful.

As reported in [25], DHC always has better performance than its alternative without the communication unit. Hence, it shows that our method can serve the non-communicating scenarios when SACHA can outperform DHC and thus DHC without communication. Besides, DHC can essentially be viewed as training our communication-based model without the attention block via the independent Q-learning. Therefore, the fact that SACHA(C) has greater performance than DHC demonstrates the strength of the heuristic-based attention mechanism and the multi-agent learning framework. Overall, our methods have a better

Table 3.2: Solution quality for different MAPF methods.

Map	Agents	Average Step Per Agent							
		CBS (120s)	ODrM* (20s)	wPBS (120s)	PRI MAL	DHC	DCC	SAC HA	SAC HA(C)
random32	4	21.82	21.82	22.90	32.96	35.70	32.83	**29.93**	31.03
	8	21.38	21.37	46.06	38.62	42.64	39.56	**36.34**	38.30
	16	31.16	31.26	172.12	45.12	48.67	43.56	41.71	**41.30**
	32	133.86	199.47	246.61	50.34	52.17	56.11	50.26	**47.72**
	64	251.30	~~256.00~~	~~256.00~~	69.40	**66.05**	88.79	76.47	74.48
random64	4	42.94	42.95	48.14	67.82	71.04	70.80	**65.47**	67.10
	8	42.74	42.80	84.52	74.68	82.43	88.94	**70.49**	72.38
	16	51.51	51.52	154.47	89.22	94.22	102.27	83.74	**82.17**
	32	94.36	136.67	222.08	98.02	103.05	126.71	95.67	**93.08**
	64	234.66	251.65	~~256.00~~	105.12	120.68	154.72	99.02	**96.42**
den312d	4	51.74	51.76	69.32	196.54	86.56	82.99	**78.33**	81.43
	8	55.50	78.74	116.32	245.02	100.70	97.95	**84.24**	89.73
	16	118.97	186.44	208.28	~~256.00~~	109.24	108.29	97.86	**96.74**
	32	251.86	~~256.00~~	248.06	~~256.00~~	124.38	119.15	111.28	**104.30**
	64	~~256.00~~	~~256.00~~	~~256.00~~	~~256.00~~	153.17	145.21	**140.79**	142.97
warehouse	4	77.79	77.79	104.41	355.80	146.12	135.89	**131.43**	134.59
	8	83.48	100.37	170.46	451.82	198.82	169.50	**164.83**	166.72
	16	81.64	133.59	340.18	492.04	281.37	208.72	**192.30**	198.72
	32	262.15	417.22	~~512.00~~	505.58	432.28	**335.81**	370.65	354.33
	64	494.93	~~512.00~~	~~512.00~~	~~512.00~~	~~512.00~~	473.92	449.83	**437.29**

chance of solving given MAPF instances, with and without communication, and among those solved instances, they result in solutions with better quality.

3.7 Summary

In this chapter, we introduced SACHA, a novel approach for learning cooperative policies for MAPF and potentially other MARL problems in partially observable environments. SACHA combines the multi-agent soft actor-critic that maximizes both expected reward and entropy with heuristic-based attention mechanisms that enhance the network architectures of both the actor and the critic. Specifically, we proposed to augment each agent's local observation with heuristic guidance from other agents and to use an attention module that learns to focus on the most relevant information for each agent to avoid collisions and achieve the goal. To the best of our knowledge, we are the first to apply the soft actor-critic with a novel agent-centered critic to homogeneous MARL settings with partial observability and to incorporate heuristic guidance and attention in the agent's policy network. We evaluated our method on various MAPF benchmarks and showed that it outperforms existing baselines for almost all the cases in terms of success rate and solution quality.

Chapter 4

Bi-Objective Cooperation

4.1 Introduction

Multi-Agent Path Finding (MAPF) [38] is a widely used technique in various multi-agent systems to find collision-free paths for agents in a shared environment. Applications include warehouse management [52], airport surface operations [29], video games [24], and other multi-agent systems [10]. Additionally, many of these applications require agents to adhere closely to a designated formation to accomplish collaborative tasks or maintain an efficient communication network. For example, in warehouse logistics, multiple robots/vehicles are required to collaborate in transporting large objects. Maintaining a specific formation is critical to optimizing transport efficiency or ensuring reliable communication. Moreover, in video gaming or military strategy simulations, game characters or army personnel must move in formations to safeguard vulnerable members.

To tackle this challenge, [17] has formalized the bi-objective problem of Moving Agents in Formation (MAiF) that combines these two tasks and proposed a centralized MAiF planner based on the leader-follower scheme and a search-based MAPF algorithm. However, existing MAiF planners work only in a centralized setting and do not apply to practical scenarios where agents do not fully observe the environment. Furthermore, centralized MAiF planners suffer from a huge computational burden as the number of agents increases and are thus not suitable for planning in real time. Additionally, the only scalable MAiF planner, SWARM-MAPF [17], does not have the flexibility to adjust to the particular preferences between two objectives since it balances the two objectives only by setting the makespan allowance between two sets of heuristically determined waypoints, thus not guaranteed to optimize targeted preference. We propose a novel approach to learning a general MAiF solver for decentralized settings that can directly adapt to various preferences of the two objectives.

In the MAPF literature, reinforcement learning and imitation learning [41] have been introduced to solve MAPF in decentralized settings [34, 21, 25]. However, most learning-based MAPF solvers learn one homogeneous policy for any set of agents that treats nearby agents as part of the environment. This learning scheme does not translate seamlessly to decentralized MAiF. Unlike MAPF where the joint action cost can be directly decomposed to action costs of individual agents, the formation in MAiF is determined by the joint state of all agents at any given time. Each agent is thus required to not only avoid colliding with other agents but coordinate with them to maintain proximity to the desired formation. The dimension of the joint state space grows exponentially with the number of agents, incapacitating the scalability. Besides, trading off two objectives merely under partial observation and limited communication make this task even more difficult.

In this work, we formalize the decentralized MAiF as a bi-objective multi-agent reinforcement learning task. The major contributions of this work are as follows. We design a practical learning formalization for MAiF, including specifications for observations, actions, rewards, and inter-agent communication. To address the aforementioned challenges of MAiF, we propose a novel approach called **Mean Field Control with Envelop *Q*-learning** (MFC-EQ), a multi-agent reinforcement learning technique that optimizes towards any linear combination of two objectives for any number of agents, ensuring a stable and efficient learning process. MFC-EQ leverages mean field control to approximate the collective dynamics of the agents, treating the interaction of each agent within the formation as influenced by the collective effect of others. This design choice facilitates seamless scalability to large-scale instances. Furthermore, MFC-EQ extends envelope Q-learning to a multi-agent setting, enabling the learning of a universal preference-agnostic model adaptable to any linear combinations of the two objectives. To evaluate our method empirically, we extensively test MFC-EQ across various MAiF instances. Our results substantiate that MFC-EQ consistently produces solutions that dominate those generated by several centralized MAiF planners and scales up well to large numbers of agents without long planning time. Additionally, the learned policy of MFC-EQ can directly adapt to more challenging tasks, including dynamically changing desired formations, which proves to be difficult for centralized MAiF planners.

4.2 Bi-Objective Optimization

We then formulate the goal of this bi-objective optimization problem. Each MAiF solution is evaluated as (v, w), where v denotes its makespan and w denotes its average formation deviation per agent. We first define dominance. We say $\boldsymbol{r} = (v, w)$ dominates $\boldsymbol{r}' = (v', w')$, denoted as $\boldsymbol{r} \preceq \boldsymbol{r}'$, iff $v \leq v'$ and $w \leq w'$. A solution is Pareto-optimal if and only if there does not exist any solution that can dominate it. The Pareto-optimal frontier is a set of all Pareto-optimal solutions. In the MAiF setting, we are also interested in evaluating each solution r by a scalar function $f_{\boldsymbol{\omega}}(\boldsymbol{r}) = \boldsymbol{\omega}^\top \boldsymbol{r}$, where $\boldsymbol{\omega} \in \Omega$ is the linear preference and Ω is the set of all possible preferences. we let $\boldsymbol{\omega} = (\lambda, 1-\lambda)^\top$ where $0 \leq \lambda < 1$. Our goal is to find the convex convergence set (CSS). The CSS is a subset of the Pareto-optimal frontier, where for each solution in CSS, there exists a preference $\boldsymbol{\omega}$ such that it minimizes $f_{\boldsymbol{\omega}}$ among

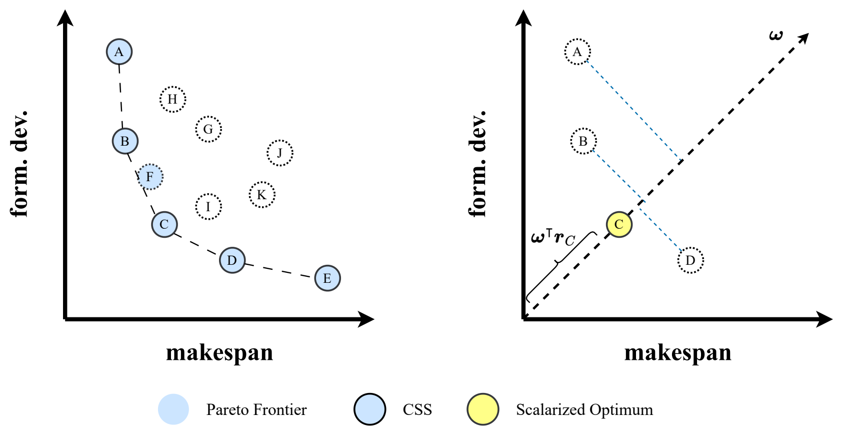

Figure 4.1: Illustration of bi-objective optimums.

all possible solutions. Intuitively, as shown in Fig. 4.1, we can regard the scalar function as a projection to the preference $\boldsymbol{\omega}$. For example, the solution C belongs to CSS since it has the smallest projection into $\boldsymbol{\omega}$ compared to others.

4.3 Related Work

We now discuss related work on mean field reinforcement learning and multi-objective reinforcement learning.

4.3.1 Mean Field Reinforcement Learning

Inspired by the mean field theory [37] from the physics world, the mean field reinforcement learning has been proposed in [54] which estimates the dynamic within the entire group of agents as the interaction between each agent and the mean effect of all other agents as a whole. As the dimension of the mean effect is independent of the number of agents, this method does not suffer from the curse of dimensionality, providing a general framework for large-scale multi-agent tasks. This method has been extended to the partially observable stochastic settings [40], which utilizes certain distributions to sample agents' actions without the necessity of observing them. The sampling process only serves stochastic games, which does not apply to our task. This mean field framework has also been used to solve multi-type multi-agent tasks [39], where agents are categorized into different types, and a set of mean effects is considered to reflect various types of agents.

4.3.2 Multi-Objective Reinforcement Learning

There exist three major categories of multi-objective reinforcement learning methods. Single-policy methods [9, 27] convert the multi-objective problem into a single-objective optimization by using linear or non-linear functions, but these methods cannot manage unknown preferences. Multi-policy methods [30, 32, 45] update on a set of policies to approximate the real Pareto-optimal frontiers, which requires immense computational resources. These methods are only applicable to problems with limited state and action space. The policy-adaptation methods either train a meta-policy that adapts to different preferences on the fly [6] or learn a policy that conditions on different preference weights [5, 1, 53]. The Envelop Q-learning [53] has been proposed to increase sample efficiency by introducing a novel envelop operator for updating the multi-objective Q-function, which has become a standard way to tackle multi-objective problems with linear preferences.

4.4 Environment and Model Design

In this section, we show how we design the learning framework for decentralized MAiF. We first design the learning environment with agents' observation, communication, action, and reward functions. Then, we elaborate on the bi-objective multi-agent learning process based on the mean field theory and the envelope Q-learning.

Observation

As most research in the MAPF community [38], we study our problem in the 2-dimensional 4-neighbor grids. To mimic many real-world robotics applications where robots have limited visibility and sense range, each agent, in our grid world, can only observe its field of view (FOV), represented by its surrounding $\mathcal{L} \times \mathcal{L}$ area. Each agent's observation is represented by 3-channel feature maps $\mathcal{F} \in \mathbb{R}^{\mathcal{L} \times \mathcal{L} \times 3}$. The first two channels indicate obstacles and other neighboring agents' positions. Inspired by some decentralized MAPF solvers [21, 25], the third channel encompasses the heuristic information where each grid in the FOV is assigned a value proportional to the short-path distance from that to the agent's goal.

Action

In 4-neighbor grids, agents can only travel to their cardinally adjacent grids for each step. The action taken by agent i at time t, denoted by $a_t^i \in \mathbb{R}^5$, is a 5-dimensional one-hot vector with each dimension representing one action from $\{up, down, left.right, wait\}$. The first four actions take agents to another location and their observation will shift accordingly. The last action is to have the agent wait at its current location, and it is especially crucial for formation control as one may have the choice for other agents to catch up for lower formation deviation.

Multi-Agent Communication

To keep the desired formation, agents not only need to communicate with nearby agents inside FOVs but also have to reach agents outside them. We specifically design the communication message so that it can pass along critical information under low communication bandwidth.

As in many real-world robot applications, each agent can only access the pairwise relative positions between other agents and itself. Assume that the current formation at time step t is $\ell_{\boldsymbol{p}} = \langle \boldsymbol{p}^1, \ldots, \boldsymbol{p}^M \rangle$ and the desired formation is $\ell_{\boldsymbol{g}} = \langle \boldsymbol{g}^1, \ldots, \boldsymbol{g}^M \rangle$. We define the relative position between agent i and j as $\boldsymbol{p}^{i,j} = \boldsymbol{p}^j - \boldsymbol{p}^i$ (resp., $\boldsymbol{g}^{i,j}$). Agent i receives $\{\boldsymbol{p}^{i,j}\}_{j \in [M]}$ in real time, and it holds the information of the relative positions in the goal formation, $\{\boldsymbol{g}^{i,j}\}_{j \in [M]}$, which can be calculated before execution. We show that, only with this information, even without knowing the agent's whereabouts, it can still calculate the formation deviation. As defined in Eq. (2.1), $\mathscr{F}(\ell_{\boldsymbol{p}}, \ell_{\boldsymbol{g}}) = \min_{\boldsymbol{\Delta}} \sum_{m=1}^{M} \|\boldsymbol{p}^m - (\boldsymbol{g}^m + \boldsymbol{\Delta})\|_1 = \sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^m - \boldsymbol{g}_n^m) - \boldsymbol{\Delta}_n|$ where $\boldsymbol{\Delta}_n$ is the median of $\{\boldsymbol{p}_n^m - \boldsymbol{g}_n^m\}_{m \in [M]}$. Recall that d is the dimension of agents' coordinates. It is easy to verify that $\mathscr{F}(\ell_{\boldsymbol{p}}, \ell_{\boldsymbol{g}})$ is also equal to $\sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^m - \boldsymbol{g}_n^m - \boldsymbol{C}_n) - \boldsymbol{\Delta}'_n|$ where $\boldsymbol{C}$ is any constant d-dimensional vector and $\boldsymbol{\Delta}'_n$ is the median of $\{\boldsymbol{p}_n^m - \boldsymbol{g}_n^m - \boldsymbol{C}_n\}_{m \in [M]}$, as all the values and the median are shifted by the same margin. Let $\boldsymbol{C} = \boldsymbol{p}^i - \boldsymbol{g}^i$. We can rewrite the definition of the formation deviation only using the relative position. $\sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^m - \boldsymbol{g}_n^m) - \boldsymbol{\Delta}_n| = \sum_{m=1}^{M} \sum_{n=1}^{d} |[(\boldsymbol{p}_n^m - \boldsymbol{p}_n^i) - (\boldsymbol{g}_n^m - \boldsymbol{g}_n^i)] - \boldsymbol{\Delta}_n^*| = \sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^{i,m} - \boldsymbol{g}_n^{i,m} - \boldsymbol{\Delta}_n^*|$ where i is the index of the observing agent and $\boldsymbol{\Delta}_n^*$ is the median of $\{\boldsymbol{p}_n^{i,m} - \boldsymbol{g}_n^{i,m}\}_{m \in [M]}$. Therefore, agent i can calculate the formation deviation merely based on the relative positions, which happens at each decentralized agent during execution with the time complexity of only $\mathcal{O}(d \cdot M)$.

We also mentioned that we can infer the mean action based on the relative positions. Given $\boldsymbol{p}^{i,j}(t)$ and $\boldsymbol{p}^{i,j}(t+1)$, we can get that $\boldsymbol{p}^{i,j}(t+1) - \boldsymbol{p}^{i,j}(t) = (\boldsymbol{p}^j(t+1) - \boldsymbol{p}^j(t)) - (\boldsymbol{p}^i(t+1) - \boldsymbol{p}^i(t)) = \boldsymbol{a}^j(t) - \boldsymbol{a}^i(t)$. Hence, agent i can calculate agent j's action by $\boldsymbol{a}^j(t) = \boldsymbol{p}^{i,j}(t+1) - \boldsymbol{p}^{i,j}(t) + \boldsymbol{a}^i(t)$. Therefore, we pass this to a linear layer to get the relative position encoding. Besides, each agent can infer other agents' actions by simply comparing relative positions in two consecutive time steps, which will later be used to compute the mean action.

Reward

The reward function for agent i after taking action a at time step t, $\boldsymbol{r}_t^i(s_t^i, a_t^i) \in \mathbb{R}^2$, is represented by a 2-tuple. The first element is designated for the makespan. We modify the individual cost function for makespan from DHC [25] which, instead, intends to minimize the sum of all path lengths (a.k.a., flowtime). The moving cost of agent i at time step t with action a^i is:

$$c_t^i(s^i, a^i) = \begin{cases} -0.075 & \text{collision-free } a^i \\ -0.5 & \text{collision (with obstacles or agents)} \\ 3 & \text{reach goal (first time)} \end{cases} .$$

Each collision-free action, including move (up, down, left, or right) and wait (on goal or away goal), is slightly penalized so that agents are incentivized to approach their goals

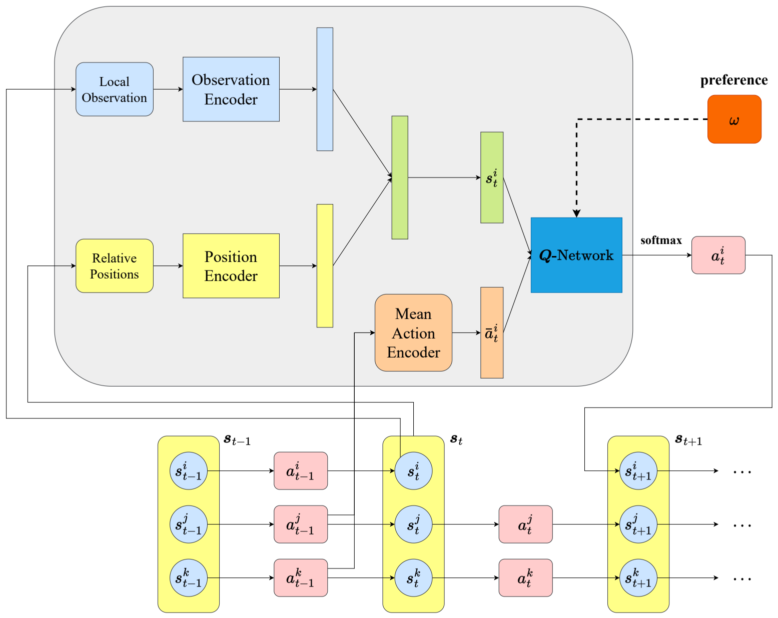

Figure 4.2: Illustration of the model architecture of MFC-EQ. The bottom demonstrates the state/observation transition in the partially observable environment. The agent's Q-network gathers information from the environment through partial observation and limited communication and chooses the next action accordingly.

as quickly as they can. The second element is for the formation deviation. As defined in Eq. (2.1), we add the individual portion of the collective formation deviation that is dedicated to agent j, namely $\mathscr{F}_t^j(\ell_t, \ell^g)$. We negate the formation deviation so that it will be minimized through maximizing rewards. Hence, the reward function can be represented as:

$$\boldsymbol{r}_t^j(s_t^j, a_t^j) = (c_t^j, -\mathscr{F}_t^j(\ell_t, \ell^g))^\top. \tag{4.1}$$

Model Architecture

Given the partially observable multi-agent environment, we further design the Q-network, whose learning algorithm will be introduced later. As in Fig. 4.2, we aim to project each agent's observations and communication messages into a corresponding action. Firstly, we feed the local observation and relative positions into two separate encoders. The observation encoder consists of several stacked convolution layers followed by linear layers. The relative position encoder includes two simple linear layers. Then, we concatenate these two encodings and forward them to another linear layer to obtain the final state representation, s_t^i, for agent i's perception at time t. We then collect other agents' actions from the previous time step to calculate the mean action. Lastly, we use stacked linear layers to project them to the Q-values which condition on the state, the action, the mean action, and the given preference. The agent will decide its next action that maximizes the Q-function produced by the Q-network.

4.5 Learning Framework

4.5.1 Mean Field and Envelop Optimality

In the rest of this section, we will discuss the details of the learning algorithm. Learning multiple policy networks, $\pi = [\pi^1, \dots, \pi^M]$, for this bi-objective multi-agent task can be extremely challenging. Therefore we simplify it by making some common assumptions.

Mean Field Approximation

The goal of MAiF is to minimize the makespan and the formation deviation. With the specifically designed reward function, the return, the discounted sum of all rewards from the initial joint state to the goal joint state, $\sum_t \sum_j \gamma^t \boldsymbol{r}_t^j(\boldsymbol{s}_t, \boldsymbol{a}_t)$, can reflect the actual values for the two objectives. Therefore, the goal of the learning is to find a set of policies to maximize the general sum of Q-values $\arg\max_{\pi^1,\dots,\pi^M} \sum_{j=1}^M \boldsymbol{\omega}^\top \boldsymbol{Q}^{\pi^j}(s^j, \boldsymbol{a})$ with the given linear preference $\boldsymbol{\omega}$. However, the dimension of $\boldsymbol{s}$ and $\boldsymbol{a}$ grows exponentially w.r.t. the number of agents, rendering it infeasible to learn efficiently.

To tackle this problem, we introduce mean field reinforcement learning. We first lay out two common assumptions of homogeneity and locality that are made in [54] and many other multi-agent reinforcement learning works. The homogeneity assumes each agent shares the same policy, meaning that $\pi^i = \pi^j$ for all $i \neq j$. The locality assumption comes from partial observability, which suggests that agents' decision-making can only depend on their visible surroundings.

Then, assuming the actions are represented by one-hot vectors, we define the mean action:

$$\bar{a}_t^j = \frac{1}{|\mathcal{N}^i|} \sum_{j \in \mathcal{N}^i} a_t^j, \; a_t^j \sim \pi^j(\cdot | s^j, \bar{a}_{t-1}^j), \tag{4.2}$$

where $\mathcal{N}^i$ denotes agent j's neighboring agents and π^j represents its policy. With the assumptions of homogeneity and locality, under certain preference $\boldsymbol{\omega}$, the local pairwise interactions can be approximated by the interplay of each agent with the mean effect from its neighbors:

$$\boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, \boldsymbol{a}_t) = \frac{1}{|\mathcal{N}^j|} \sum_{k \neq j} \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, a_t^k) = \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, \bar{a}_t^j),$$

where $\boldsymbol{a}_t$ is the joint action, a_t is the single-agent action, and $\bar{a}_t$ is the mean action. Given this approximated Q-function, we can derive the agent's policy function with the softmax parameterization:

$$\pi^j(a_t^j|s_t^j, \bar{a}_{t-1}^j) = \frac{\exp(\beta\boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, \bar{a}_{t-1}^j, \boldsymbol{\omega}))}{\sum_{a\in A^j}\exp(\beta\boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a, \bar{a}_{t-1}^j, \boldsymbol{\omega}))}, \tag{4.3}$$

where β is the Boltzmann parameter.

Bellman Optimality Operator

To extend this framework to multi-objective reinforcement learning, we modify the envelop Q-learning [53] by combining the mean field operator with the envelop optimality operator. We first condition all the Q-values on the linear preference $\boldsymbol{\omega}$, as in $\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$. As the standard Q-learning [50], we define the bi-objective multi-agent Bellman optimality operator $\mathcal{T}$ as:

$$(\mathcal{T}\boldsymbol{Q})(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\omega}) := \sum_{j=1}^{M} \boldsymbol{r}^j(s_t^j, a_t^j) + \gamma\mathbb{E}_{s_{t+1}} \sum_{j=1}^{M} \arg_{Q^j}\{\max_{\boldsymbol{\omega}'\in\Omega}\max_{a^j\in A^j} \boldsymbol{\omega}^\top \boldsymbol{Q}^j(s_{t+1}^j, a^j, \bar{a}_{t+1}^j, \boldsymbol{\omega}')\}, \tag{4.4}$$

where $\arg_{Q^j}$ takes out $\boldsymbol{Q^j}$ that maximizes $\boldsymbol{\omega}^\top\boldsymbol{Q^j}$. This operator resembles the Bellman optimality operator in the standard Q-learning for single-agent RL and provides the temporal difference (TD) target. The difference is that it also optimizes over the parameter of preference $\boldsymbol{\omega}$. By maximizing $\boldsymbol{\omega'}$ over the next state and its onward trajectory, this approach provides an optimistic perspective for its future rewards. Iteratively applying this operator to the Q-function, we will be able to reach the convergence of the near-optimal Q-function, which has been proven in [53].

4.5.2 Double Q-learning

We design our learning algorithm based on the double Q-learning [14] with two different loss functions and the target network. Algorithm 1 presents the detailed learning framework. During the rollout phase (Line 5-10), we sample the transitions in the multi-agent environment with the homogeneous policy. After we obtain enough transitions in the replay buffer, we enter the learning phase (Line 11-14). Given a mini-batch of N transitions and N_ω preferences, we can estimate the TD target $\boldsymbol{y} = (\mathcal{T}\boldsymbol{Q})(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$ via Eq. (4.4). The first loss function can be computed as the L_2-norm of the multi-objective TD:

$$L_A(\theta) = \mathbb{E}_{\boldsymbol{s},\boldsymbol{a},\boldsymbol{\omega}}\Big[\|\boldsymbol{y} - \sum_{j=1}^{M}\boldsymbol{Q}_\theta(s^j, a^j, \bar{a}^j, \boldsymbol{\omega})\|_2^2\Big].$$

Although this loss function is close to the true expected return, the non-smooth surface makes the learning process difficult in the early steps. We combine this with another

Algorithm 1: Mean Field Control with Envelop Q-learning

```
1  Initialize the $Q$-network $\boldsymbol{Q}_\theta$ and the target $Q$-network $\boldsymbol{Q}_{\bar{\theta}}$
2  Initialize the replay buffer $\mathcal{D}$ and set $\zeta = 0$
3  for episode $= 1, \dots, E$ do
4      Initialize $\bar{a}_0^j$ for all $j \in [M]$
5      for $t = 1, \dots, T_{\max}$ do
6          Sample $a_t^j$ $\epsilon$-greedily from $\boldsymbol{Q}_\theta$ by Eq. (4.3) for all $j \in [M]$
7          Compute new mean actions $\bar{a}_t^j$ by Eq. (4.2) for all $j \in [M]$
8          Take the joint action $\mathbf{a}_t = [a_t^1, \dots, a_t^M]$ from the state $\mathbf{s}$ to the next state $\mathbf{s}_{t+1}$
9          Compute the reward $\mathbf{r}_t = [r^1, \dots, r^M]$ by Eq. (4.1)
10         Store the transition, $\langle \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}, \bar{\boldsymbol{a}} \rangle$, into $\mathcal{D}$, where $\bar{\boldsymbol{a}}_t = [\bar{a}_t^1, \dots, \bar{a}_t^M]$ is
             the collection of mean actions
11     if update then
12         Sample $N$ transitions from $\mathcal{D}$ and $N_{\boldsymbol{\omega}}$ preferences from $\Omega$
13         Compute the TD target using the operator in Eq. (4.4)
14         Update $Q_\theta$ by minimizing the loss from Eq. (4.5)
15     Update $\boldsymbol{Q}_{\bar{\theta}}$ with the learning rate $\alpha$: $\bar{\theta} \leftarrow \alpha\theta + (1-\alpha)\bar{\theta}$
16     Increase $\zeta$ along the predefined homotopy path
```

additional loss function with the projected temporal difference:

$$L_B(\theta) = \mathbb{E}_{\boldsymbol{s},\boldsymbol{a},\boldsymbol{\omega}}\Big[|\boldsymbol{\omega}^\top(\boldsymbol{y} - \sum_{j=1}^{M} \boldsymbol{Q}_\theta(s^j, a^j, \bar{a}^j, \boldsymbol{\omega}))|\Big].$$

$L_A(\theta)$ provides a closer estimation of the true Q-function since it evaluates the Q-function w.r.t. the optimal frontier which could contain a large number of solutions and result in difficulties for optimization. $L_B(\theta)$, on the other hand, makes the landscape of optimization smooth and easy to optimize. We train the Q-network via the homotopy optimization [51] based on the combination of these two loss functions:

$$L(\theta) = (1-\zeta)L_A(\theta) + \zeta L_B(\theta), \tag{4.5}$$

where we gradually increases ζ from 0 to 1 exponentially as the learning proceeds.

4.6 Empirical Evaluation

In this section, we implement MFC-EQ and experimentally evaluate it with other methods on a server equipped with Intel 2.3GHz 16-Core CPUs and NVIDIA A40 GPUs.

4.6.1 Experimental Setups

We use 4-neighbor grids with 2 obstacle-free corners in the top-left and the bottom-right. The default obstacle density for grids outside these two corners is set to be 10%. The agents start at the top-left corner and travel toward the bottom-right corner. The formation in the

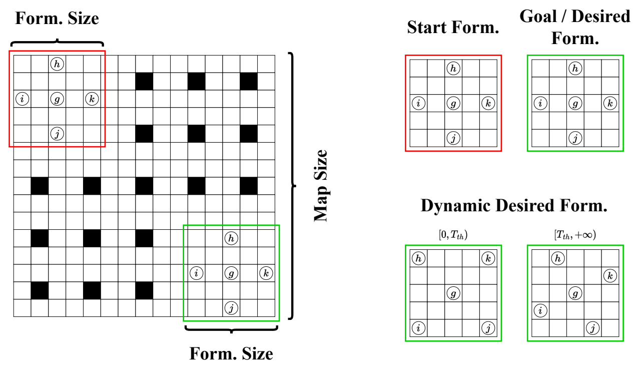

Figure 4.3: Demonstration of experiment environments.

goal position represents the desired formation. We refer to the size of grids as map size and the size of corners as formation size. For each data point, we averaged over 100 samples by crossing 10 random maps and 10 random formations.

As shown in Fig. 4.3, we conduct our experiment in random grids with various map sizes and formation box sizes. We place two obstacle-free corners. The top-left corner is the start locations, and the bottom-right corner is the goal locations which makes the desired formation. Other grids are blocked with 10% probability uniformly at random. In our method, all the decentralized agents have 9×9 FOV. We vary the number of agents, the map size, and the formation size to extensively test the performance of these methods in different environments.

4.6.2 Centralized Methods

We compare our method with several methods which have to plan all paths on centralized servers before execution.

Scalarized Prioritized Planning (SPP)

Since it is NP-hard to solve this problem optimally, we come up with an efficient yet sub-optimal baseline based on the prioritized planning algorithm [36]. We first give each agent a unique priority, and in that priority order, a low-level A* search will be invoked to plan the path from the start location to the goal location while respecting the already planned paths of all agents with higher priorities. The low-level A* search uses a scalarized f-value for each state, which is a mix of the makespan f-value, f_{MS}, and the formation deviation f-value, f_{FD}:

$$f(n) = \underbrace{\lambda \left[cost(\boldsymbol{v}^i) + dist(\boldsymbol{v}^i, g^i) \right]}_{f_{MS}(n)} + (1-\lambda) \underbrace{\sum_{t=1}^{cost(\boldsymbol{v}^i)} \mathscr{F}_P^i(t_n)}_{f_{FD}(n)},$$

where $\mathscr{F}^i_P(t_n)$ is the partial formation deviation among agent i and all other agents with higher priorities. This baseline is not complete but, in most cases, can find a possible solution much more quickly, albeit the solutions usually have poor quality, especially in congested environments with large numbers of agents. Moreover, this planner, unlike SWARM-MAPF, can target any given linear preference. In experiments, we use the above scalarized f-value, and the weights for makespan and formation deviation are set to λ and $1-\lambda$, respectively. We vary λ from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ to test its performance under different linear preferences.

SWARM-MAPF (SWARM)

The most effective centralized method, SWARM-MAPF, has been proposed in [17], which combines the swarm-based formation control with the conflict-based MAPF algorithms. The SWARM-MAPF is a two-phase algorithm. In Phase 1, it first calculates the lower bound of the makespan $B = \max_{1 \le i \le M} \mathit{dist}(s_i, g_i)$. Given a user-provided parameter $w \ge 1$, SWARM-MAPF selects a leader from the group of agents such that its path, whose length is bounded by wB, can be sufficiently far away from the obstacles and thus others agents can preserve their formation as much as they can. In Phase 2, it will invoke the modified conflict-based search [35] (CBS-M) to minimize the makespan and replanning some critical segments. This planner is complete and suboptimal, but it cannot specifically target any given preference, since we cannot control the trade-off between two objectives based on the parameter w.

Joint State A* (JSA*)

The joint state A* [33] directly applies the ϵ-constraint search algorithm [13] in the joint state space. The joint state assigns all agents a set of different locations. The operator assigns each agent a set of non-colliding move or wait actions. The OPEN list sorts nodes based on makespan, while the FOCAL list breaks ties based on the formation deviation. Since the joint state space grows exponentially w.r.t the number of agents, this method can only be applied to instances with relatively small agents (less than 5 agents in our setups). By varying the ϵ in the focal search, this method is guaranteed to find the Pareto-optimal frontier, albeit significantly slower.

4.6.3 Main Results

Linear Preferences

We first evaluate the ability of our learned Q-Network to adapt to different preferences. We use the environment that has 16 agents with 48×48 map size and 9×9 formation size. We test different preferences $\boldsymbol{\omega} = (\lambda, 1-\lambda)^\intercal$ by varying λ from 0.1 to 0.9. We also evaluate each result under different `MIX`(λ) objectives by varying λ from the same set of values. Table 4.1 provides 5 different solutions, and every `MIX` column marks the solution that minimizes

Table 4.1: Results for MFC-EQ with different preferences evaluated by different objectives.

$\boldsymbol{\omega}(\lambda)$	Make-span	Form. Dev.	`MIX`(0.1)	`MIX`(0.3)	`MIX`(0.5)	`MIX`(0.7)	`MIX`(0.9)
0.1	106.33	14.67	**23.84**	42.17	60.50	78.83	97.16
0.3	101.14	15.37	23.95	**41.10**	58.26	75.41	92.56
0.5	98.64	16.84	25.02	41.38	**57.74**	74.10	90.46
0.7	96.74	19.16	26.92	42.43	57.95	**73.47**	88.98
0.9	96.42	21.75	29.22	44.15	59.09	74.02	**88.95**

the projection onto that particular preference. As we can observe, all `MIX`(λ) objectives are minimized by feeding the corresponding preference $\boldsymbol{\omega}(\lambda)$ into the Q-network. This suggests that the learned Q-network using MFC-EQ can adapt to different preferences and produce multiple solutions that fit the given preferences.

Number of Agents

We evaluate our methods under different numbers of agents in different map sizes and compare the results with centralized baselines. The λ is set to 0.5 for SPP and MFC-EQ. The w is set to 1.0 for SWARM. The runtime limit is only 30 seconds for MFC-EQ and 5 minutes for SWARM and SPP. Due to the partially observable environment, our method naturally does not have perfect success rates, but they are relatively high and acceptable. As we can observe from Table 4.2, SWARM has greater performance in almost all the test cases. We also can see when projected to the demanded preference, our method can outperform SWARM in most instances. This experiment shows that our method can scale up well to instances with large numbers of agents in different sizes of maps.

Formation Size

We repeat the experiment above with various formation sizes. We choose different sizes of obstacle-free corners in which the formation is randomly generated. The larger the corner is, the more spread out the formation will be. The number of agents is fixed at 16. As shown in Table 4.3, we see that smaller formations are usually more difficult to solve, resulting in larger makespan and formation deviation. Compared to SWARM, SPP generally has a better makespan but much worse formation deviation and `MIX`. MFC-EQ solves most instances with greater solution quality in both objectives when compared to the baselines.

Dynamic Formation

We also put these methods into more challenging tests where the agents will be asked to adjust to different formations on the fly. We evaluate agents' formations with one desired formation before $T_{th} = 30$ and another different formation after that, as shown in the right-bottom of Fig. 4.3. The centralized scheme cannot handle such tasks as agents' paths will

Table 4.2: Results for MFC-EQ and centralized baselines with different numbers of agents in various sizes of grids.

		Success Rate			Makespan			Form. Dev.			`MIX`(0.5)		
Map Size	M	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ
32	10	1.00	1.00	1.00	48.30	59.32	60.24	29.79	6.07	4.35	39.05	32.70	**32.30**
×	20	1.00	0.99	0.99	49.03	63.17	60.38	32.42	12.38	10.04	40.73	37.78	**35.21**
32	30	0.79	0.96	0.90	51.54	59.09	54.59	42.25	20.64	20.32	46.90	39.87	**37.46**
48	10	1.00	0.99	0.99	80.44	98.10	88.07	53.91	8.18	11.05	67.18	53.14	**49.56**
×	20	0.95	0.99	0.96	82.07	108.84	104.28	70.44	23.70	21.49	76.26	66.27	**62.89**
48	30	0.74	0.94	0.88	84.92	101.52	107.42	96.04	36.30	37.18	90.48	**68.91**	72.30
64	10	1.00	0.99	0.99	113.38	144.54	137.14	97.92	15.00	16.43	105.65	79.77	**76.79**
×	20	1.00	0.97	0.93	114.56	156.03	141.26	113.52	33.24	28.34	114.04	94.64	**84.80**
64	30	0.22	0.98	0.90	115.59	142.65	145.51	107.64	57.31	61.43	111.62	**99.98**	103.47

Table 4.3: Results for MFC-EQ and centralized baselines under different formation sizes in various sizes of grids.

		Success Rate			Makespan			Form. Dev.			`MIX`(0.5)		
Map Size	Form Size	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ
32	7× 7	0.94	1.00	0.97	53.81	66.40	68.30	44.66	12.37	9.06	49.24	39.39	**38.68**
×	9× 9	1.00	1.00	1.00	48.56	63.20	67.33	29.34	9.10	8.72	38.95	**36.15**	38.03
32	11× 11	1.00	1.00	1.00	44.18	57.75	55.12	20.80	7.03	8.32	32.49	32.39	**31.72**
48	7× 7	0.98	1.00	0.87	86.26	110.94	107.37	82.85	19.84	22.40	84.56	65.39	**64.89**
×	9× 9	0.93	0.96	0.93	81.49	109.67	98.64	65.74	21.03	16.84	73.62	65.35	**57.74**
48	11× 11	1.00	1.00	1.00	77.06	105.06	97.26	60.65	15.12	14.08	68.86	60.09	**55.67**
64	7× 7	0.87	0.99	0.96	118.64	155.36	138.84	115.38	31.07	55.42	117.01	**93.22**	97.13
×	9× 9	1.00	1.00	0.96	113.87	153.33	133.92	108.57	25.95	33.20	111.22	89.64	**83.56**
64	11× 11	1.00	0.95	1.00	109.43	149.61	131.08	97.68	23.02	27.75	103.56	86.32	**79.42**

have to be planned before execution. In our method, we can simply notify each decentralized agent of the new goal formation which will result in different ways of calculating relative positions, and therefore the agents can adjust to the new formation seamlessly. The results are shown in Table 4.4, suggesting that our method has the flexibility to tackle changeable formations, while others result in much larger deviation.

Makespan and Formation Trade-off

We further compare our method with others under different preferences. Due to the limited scalability of FOCAL, we first use the environment with 20×20 map size, 3×3 formation size, 15% obstacle density, and 3 agents. We vary ϵ of FOCAL from 1.0 to 1.8 and w of SWARM from 1.0 to 1.6. The value of λ for $\boldsymbol{\omega}$ in MFC-EQ is varied from 0.1 to 0.9. FOCAL can provide the Pareto-optimal frontier only for small-scale instances. We then

Table 4.4: Results for MFC-EQ and centralized baselines with dynamic formation.

	Success Rate			Makespan			Form. Dev.			MIX(0.5)		
M	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ
10	1.00	0.98	0.96	48.19	59.00	56.33	127.90	172.40	104.61	88.05	115.70	**80.47**
15	1.00	1.00	1.00	48.29	63.85	57.56	132.71	210.82	114.33	90.50	137.34	**85.95**
20	0.97	1.00	1.00	49.64	63.60	59.07	141.18	208.28	118.42	95.41	135.94	**88.75**
25	0.72	1.00	1.00	50.56	62.58	61.29	149.61	204.33	123.20	100.09	133.46	**92.25**
30	0.90	0.98	0.93	50.00	59.31	62.50	146.82	187.08	129.74	98.41	123.20	**96.12**
35	0.48	0.94	0.87	51.75	57.87	64.71	163.40	189.64	133.07	107.58	123.76	**98.89**
40	0.25	0.81	0.74	52.68	54.12	65.29	168.64	165.05	137.33	110.66	109.59	**101.31**

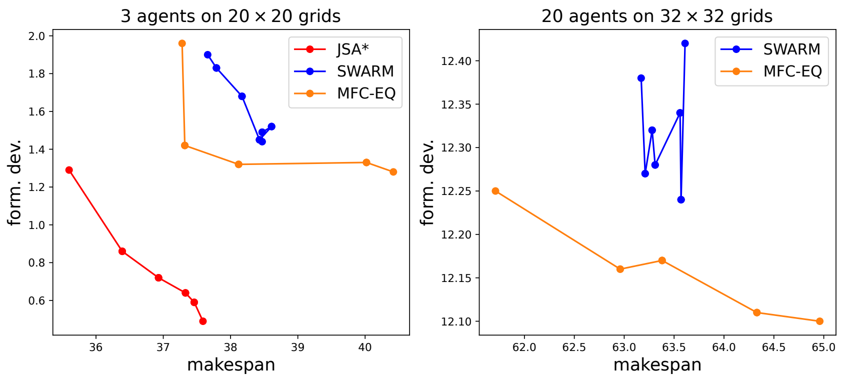

Figure 4.4: Trade-off of makespan and formation deviation.

repeat this experiment in larger instances with 32×32 map size, 9×9 formation size, and 20 agents. Fig. 4.4 shows the results. Although SPP is also tested, it only gives solutions that have near-optimal makespan but significantly larger formation deviation than the shown results. In large-scale cases, SWARM tends to fluctuate, meaning that, even given more makespan allowance, it may result in solutions with worse formation deviation. The envelope generated by our method is near-convex and can cover all solutions from SWARM, albeit still suboptimal. It also has a wider range of makespan with greater solution variety.

4.7 Summary

We proposed MFC-EQ, a general Q-learning framework for solving decentralized MAiF with partial observation and limited communication effectively. MFC-EQ utilizes the mean field approximation to simplify the complex multi-agent interaction and employs the envelop Q-learning to enable the adaptability to various linear preferences for this bi-objective task. Empirical results demonstrate that MFC-EQ outperforms existing centralized baselines in most cases and is more versatile in handling dynamically changing desired formations.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In many real-world applications in multi-agent systems, groups of agents are required to find their collision-free paths from start locations to goal locations. While search-based MAPF algorithms have been developed in the AI community, they do not apply to agents with limited sensing and communication capabilities and do not scale well to large numbers of agents. We consider a more practical setting where decentralized agents can only decide their sequential actions based on their partial observation of the environment. In this setting, we investigate two specific tasks, one is single-objective cooperation in partially observable MAPF and the other is bi-objective cooperation in partially observable MAiF. For the first task, we propose SACHA, a novel multi-agent soft actor-critic framework with attention mechanisms. Using this framework, we can learn a policy for agents that can generalize among various environments (Chapter 3). As for the second task, we propose MFC-EQ, a combination of mean field RL and multi-objective RL for tackling this bi-objective multi-agent task under any linear preference (Chapter 4). We provide theoretical analysis to show the effectiveness of our methods. Empirical evaluation suggests that our methods outperform centralized and decentralized baselines in most given instances over various environments. Moreover, the proposed methods are not limited to path planning and have the potential to extend to other tasks in multi-agent systems.

5.2 Future Work

We believe that our methods can be extended to other domains in multi-agent systems. We list a few promising directions below.

- In single-objective cooperation, it could be interesting to consider more universal schemes that can improve generalizability among different environments. For example, we could apply meta-learning, which obtains a pre-trained model at first and fine-tunes it when given the specific environment.

- In bi-objective cooperation, we could apply other multi-objective RL algorithms that do not depend on the linear combinations of rewards. Instead, we could optimize over the multi-objective rewards and directly approach the Pareto frontier.
- For path planning in multi-agent cooperation, we could consider more sophisticated types of cooperation, such as task assignment and multi-type agents.

We will leave these directions for future work.

Bibliography

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*, pages 11–20. PMLR, 2019.

[2] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. Intractability of time-optimal multirobot path planning on 2d grid graphs with holes. *IEEE Robotics and Automation Letters*, 2(4):1941–1947, 2017.

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[5] Andrea Castelletti, Francesca Pianosi, and Marcello Restelli. Multi-objective fitted q-iteration: Pareto frontier approximation in one single run. In *2011 International Conference on Networking, Sensing and Control*, pages 260–265. IEEE, 2011.

[6] Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Meta-learning for multi-objective reinforcement learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 977–983. IEEE, 2019.

[7] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.

[8] 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.

[9] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. Multi-criteria reinforcement learning. In *ICML*, volume 98, pages 197–205, 1998.

[10] Avinash Gautam and Sudeept Mohan. A review of research in multi-robot systems. In *ICIIS*, pages 1–5. IEEE, 2012.

[11] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[12] 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.

[13] Yacov Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics*, (3):296–297, 1971.

[14] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[15] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pages 2961–2970. PMLR, 2019.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Jiaoyang Li, Kexuan Sun, Hang Ma, Ariel Felner, TK Kumar, and Sven Koenig. Moving agents in formation in congested environments. In *Proceedings of the International Symposium on Combinatorial Search*, volume 11, pages 131–132, 2020.

[18] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11272–11281, 2021.

[19] 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.

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

[21] 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.

[22] 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.

[23] Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7643–7650, 2019.

[24] Hang Ma, Jingxing Yang, Liron Cohen, TK Kumar, and Sven Koenig. Feasibility study: Moving non-homogeneous teams in congested video game environments. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 13, pages 270–272, 2017.

[25] 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.

[26] 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.

[27] Shie Mannor and Nahum Shimkin. The steering approach for multi-criteria reinforcement learning. *Advances in Neural Information Processing Systems*, 14, 2001.

[28] 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.

[29] Robert Morris, Corina S Pasareanu, Kasper Søe Luckow, Waqar Malik, Hang Ma, TK Satish Kumar, and Sven Koenig. Planning, scheduling and monitoring for airport surface operations. In *AAAI Workshop: Planning for Hybrid Systems*, pages 608–614, 2016.

[30] Sriraam Natarajan and Prasad Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 601–608, 2005.

[31] Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722, 2023.

[32] Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330. IEEE, 2014.

[33] Judea Pearl and Jin H Kim. Studies in semi-admissible heuristics. *IEEE transactions on pattern analysis and machine intelligence*, (4):392–399, 1982.

[34] 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.

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

[36] David Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[37] H Eugene Stanley. *Phase transitions and critical phenomena*, volume 7. Clarendon Press, Oxford, 1971.

[38] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, pages 151–158, 2019.

[39] Sriram Ganapathi Subramanian, Pascal Poupart, Matthew E Taylor, and Nidhi Hegde. Multi type mean field reinforcement learning. *arXiv preprint arXiv:2002.02513*, 2020.

[40] Sriram Ganapathi Subramanian, Matthew E Taylor, Mark Crowley, and Pascal Poupart. Partially observable mean field reinforcement learning. *arXiv preprint arXiv:2012.15791*, 2020.

[41] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[42] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[43] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[44] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. *Readings in Agents*, pages 487–494, 1997.

[45] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[46] 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.

[47] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. Cobots: Robust symbiotic autonomous mobile service robots. In *Twenty-fourth international joint conference on artificial intelligence*. Citeseer, 2015.

[48] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 3260–3267. IEEE, 2011.

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

[50] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[51] Layne T Watson and Raphael T Haftka. Modern homotopy methods in optimization. *Computer Methods in Applied Mechanics and Engineering*, 74(3):289–305, 1989.

[52] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

[53] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32, 2019.

[54] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.

[55] Jingjin Yu and Steven LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 1443–1449, 2013.