2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

MFC-EQ: Mean-Field Control with Envelope Q-learning for Moving Decentralized Agents in Formation

Qiushi Lin and Hang Ma

***Abstract*— We study a decentralized version of Moving Agents in Formation (MAiF), a variant of Multi-Agent Path Finding aiming to plan collision-free paths for multiple agents with the dual objectives of reaching their goals quickly while maintaining a desired formation. The agents must balance these objectives under conditions of partial observation and limited communication. The formation maintenance depends on the joint state of all agents, whose dimensionality increases exponentially with the number of agents, rendering the learning process intractable. Additionally, learning a single policy that can accommodate different linear preferences for these two objectives presents a significant challenge. In this paper, we propose Mean-Field Control with Envelop Q-learning (MFC-EQ), a scalable and adaptable learning framework for this bi-objective multi-agent problem. We approximate the dynamics of all agents using mean-field theory while learning a universal preference-agnostic policy through envelop Q-learning. Our empirical evaluation of MFC-EQ across numerous instances shows that it outperforms state-of-the-art centralized MAiF baselines. Furthermore, MFC-EQ effectively handles more complex scenarios where the desired formation changes dynamically—a challenge that existing MAiF planners cannot address.**

I. Introduction

Multi-Agent Path Finding (MAPF) [1], [2] is a well-studied problem in various multi-agent systems, aiming to find collision-free paths for agents in a shared environment. Its applications include warehouse management [3], airport surface operations [4], video games [5], and other multi-agent systems [6]. Many of these applications require agents to adhere closely to a designated formation to accomplish collaborative tasks or ensure an efficient communication network. For example, in warehouse logistics, multiple robots or vehicles must collaborate to transport large objects, where maintaining a specific formation is critical for optimizing transport efficiency or ensuring reliable communication. Similarly, in video gaming or military strategy simulations, game characters or army personnel must move in formations to safeguard vulnerable members.

To tackle this challenge, [7] formalized the bi-objective problem of Moving Agents in Formation (MAiF), which combines these two tasks, and proposed a centralized MAiF planner based on a leader-follower scheme and a search-based MAPF algorithm. However, existing MAiF planners are designed for centralized settings and are not applicable in practical scenarios where agents lack full observation of the environment. Additionally, centralized MAiF planners face significant computational challenges as the number of agents increases, making them unsuitable for real-time planning. Moreover, the only scalable MAiF planner, SWARM-MAPF [7], lacks the flexibility to adjust to specific preferences between the two objectives, as it balances them only by setting a makespan allowance between two sets of heuristically determined waypoints, without guaranteeing optimization toward a targeted preference. We propose a novel approach for learning a general MAiF solver suitable for decentralized settings that can directly adapt to various preferences between the two objectives.

In the MAPF literature, reinforcement learning and imitation learning have been explored to solve MAPF in decentralized settings [8], [9], [10]. 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 seamlessly translate to decentralized MAiF. Unlike MAPF where the joint action cost can be directly decomposed into action costs of individual agents, formation maintenance in MAiF depends on the joint state of all agents at any given time. Each agent must not only avoid colliding with others but also coordinate with them to maintain proximity to the desired formation. The dimensionality of the joint state space grows exponentially with the number of agents, which hinders scalability. Additionally, trading off the two objectives under partial observation and limited communication further complicates this task.

In this paper, we formalize decentralized MAiF as a bi-objective multi-agent reinforcement learning problem. The major contributions of our 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 **M**EAN **F**IELD **C**ONTROL WITH **E**NVELOP ***Q***-LEARNING (MFC-EQ), a multi-agent reinforcement learning technique that optimizes toward 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 combination of the two objectives. To evaluate our method empirically, we extensively test MFC-EQ across various MAiF instances.

This work was supported by the NSERC under grant number RGPIN2020-06540 and a CFI JELF award.

The authors are with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada {qiushi_lin, hangma}@sfu.ca

Our code is publicly available at https://github.com/Qiushi-Lin/MFCEQ.

Our results substantiate that MFC-EQ consistently produces solutions that surpass those generated by several centralized MAiF planners and scale effectively 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 difficult for centralized MAiF planners.

II. Problem Definition

In this section, we first describe the standard MAiF formulation using terminology that facilitates the presentation of our learning approach. We then discuss how MAiF can be extended to a partially observable environment, which is a more practical problem setting. Finally, we define relevant concepts and outline the bi-objective optimization problem.

A. Standard Formulation of MAiF

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 identified by its coordinates $\boldsymbol{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$. In this paper, superscripts represent agents' indices and boldface denotes multi-dimensional vectors. We define a set of M agents $I = \{a^i | i \in [M]\}$, where $[M] = \{1, \ldots, M\}$. Each agent has a unique start location $\boldsymbol{s}^i \in V$ and goal location $\boldsymbol{g}^i \in V$. Time is discretized, and at each time step, an agent can either wait at its current location or move from $\boldsymbol{v}$ to $\boldsymbol{v}'$, provided $(\boldsymbol{v}, \boldsymbol{v}') \in E$. We consider two types of collision between agents a^i and a^j at time step t: A vertex collision $\langle a^i, a^j, \boldsymbol{v}, t\rangle$ occurs when they occupy the same location $\boldsymbol{v}$, and an edge collision $\langle a^i, a^j, \boldsymbol{u}, \boldsymbol{v}, t\rangle$ occurs when a^i moves from $\boldsymbol{u}$ to $\boldsymbol{v}$ while a^j moves in the opposite direction.

The MAiF problem aims to find a set of M collision-free paths $\Pi = \{\Pi^i | i \in [M]\}$ as a solution, where $\Pi^i = (p^i_0, \ldots, p^i_{T^i})$ represents agent i's trajectory. Each solution is evaluated based on two objective functions, makespan and formation deviation. The makespan is defined as $T = \max_{1 \le i \le M} T^i$, representing the longest path length among all agents. The *formation* at time t can be represented as an M-tuple, $\ell(t) = \langle \boldsymbol{p}^1(t), \ldots, \boldsymbol{p}^M(t)\rangle$. The desired formation corresponds to a combination of all agents' goal locations, $\ell_{\boldsymbol{g}} = \langle \boldsymbol{g}^1, \ldots, \boldsymbol{g}^M\rangle$. Following the definition in [7], the *formation deviation* between any two formations $\ell = \langle \boldsymbol{u}^1, \ldots, \boldsymbol{u}^M\rangle$ and $\ell' = \langle \boldsymbol{v}^1, \ldots, \boldsymbol{v}^M\rangle$ indicates the least effort required to transform ℓ into ℓ', defined as:

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

where j indexes the dimension for each vector and $\Delta_j = \mathit{median}(\{\boldsymbol{u}^i_j - \boldsymbol{v}^i_j\}_{i \in [M]})$ is the median of the differences between the coordinates in the j-th dimension across all agents. The term $\mathscr{F}^i(\ell, \ell')$ denotes the component related to only agent a^i. We consider the average formation deviation per agent across all time steps, defined as $\mathscr{F}_{avg} = \frac{1}{M} \sum_{t=0}^{T} \mathscr{F}(t)$, where $\mathscr{F}(t) = \mathscr{F}(\ell(t), \ell_{\boldsymbol{g}})$, which is more convenient in our decentralized setting than the total formation deviation used in [7]. Additionally, we consider a linear combination of the two objectives:

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

where λ balances the trade-off between makespan and formation deviation.

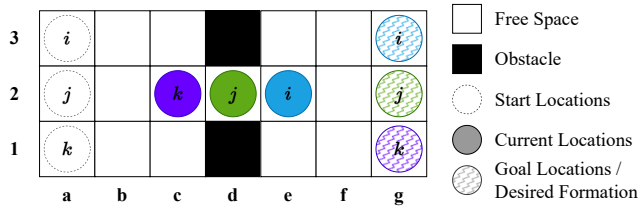

Fig. 1: Example of moving agents in formation.

Example A simple MAiF example is demonstrated in Fig. 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 $\mathscr{F}(t) = \mathscr{F}^i(t) + \mathscr{F}^j(t) + \mathscr{F}^k(t) = 2 + 0 + 2 = 4$.

B. Partially Observable Environment

In this paper, we consider a more practical problem setting where, instead of having full knowledge of the environment, each agent can only observe part of its surroundings. We first introduce the standard decentralized partially observable Markov Decision Process (Dec-POMDP) [11]. A Dec-POMDP is represented by 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. We adopt the standard Dec-POMDP framework to model our decentralized MAiF problem, assuming both the observation-transition and state-transition functions are deterministic. We also assume each agent can take an action based solely on its local observation and limited communication with others. Following existing learning-based MAPF literature [8], [10], we formalize this problem on a 2D 4-neighbor grid, though our method can be easily generalized to other settings. Partial observability restricts each agent's perception to a $\mathcal{L} \times \mathcal{L}$ square area centered on the agent, defined as its FOV.

C. Bi-Objective Optimization

We now define the goal of this bi-objective optimization problem. Each MAiF solution is evaluated as $\boldsymbol{r} = (v, w)$, where v denotes its makespan and w denotes its average formation deviation per agent. We first define dominance:

$\boldsymbol{r} = (v, w)$ dominates $\boldsymbol{r}' = (v', w')$, denoted as $\boldsymbol{r} \preceq \boldsymbol{r}'$, if and only if $v \leq v'$ and $w \leq w'$. A solution is Pareto-optimal if there does not exist any other solution that dominates it. The Pareto-optimal frontier is the set of all Pareto-optimal solutions. In the MAiF setting, we are also interested in evaluating each solution r using a scalar function $f_{\boldsymbol{\omega}}(\boldsymbol{r}) = \boldsymbol{\omega}^\top \boldsymbol{r}$, where $\boldsymbol{\omega} \in \Omega$ represents a linear preference and Ω is the given distribution over a set of possible preferences. We use $\boldsymbol{\omega} = (\lambda, 1-\lambda)^\top$ where $0 \leq \lambda < 1$.

III. Related Work

This section reviews related work on decentralized MAPF and MAiF, mean-field reinforcement learning, and multi-objective reinforcement learning.

A. Learning-Based MAPF and MAiF Solvers

Recently, reinforcement learning has been introduced to solve decentralized variants of MAPF [8], [10], [12]. These methods are designed to learn a decentralized model that can be generalized across different MAPF instances. Traditional centralized MAPF planners usually require full observation of the environment and must plan paths from scratch for each instance. In contrast, well-trained learning-based models offer the advantage of being applicable to any MAPF instances, regardless of the number of agents or the size of the environment. For decentralized MAiF, [13] explored a similar setup to our work and proposed a hierarchical reinforcement learning scheme to divide the bi-objective task into unrelated sub-tasks. However, the hierarchical learning structure hinders the learned model's ability to adapt to different preference weights between the two objectives. Additionally, the simplistic network design struggles to scale to large numbers of agents. [13] also employs a different definition of formation deviation, and for that reason, we do not compare their results with our method in Section V.

B. Mean-Field Reinforcement Learning

Inspired by mean-field theory [14] from physics, mean-field reinforcement learning has been proposed in [15] to estimate the dynamics within an entire group of agents by modeling the interaction between each agent and the mean effect of all other agents as a whole. Since the dimensionality of the mean effect is independent of the number of agents, this method avoids the curse of dimensionality, providing a general framework for large-scale multi-agent tasks. This approach has been extended to partially observable stochastic settings [16], where certain distributions are used to sample agents' actions without the necessity of observing them. However, the sampling process is suited only for stochastic games, making it inapplicable to our task.

C. Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning methods can be categorized into three major types. Single-policy methods [17], [18] convert a multi-objective problem into a single-objective optimization using linear or non-linear functions, but these methods cannot handle unknown preferences. Multi-policy methods [19], [20], [21] work by updating a set of policies to approximate the true Pareto-optimal frontiers, but these methods require immense computational resources and are only feasible for problems with limited state and action spaces. Policy-adaptation methods either train a meta-policy that adapts to different preferences on the fly [22] or learn a policy conditioned on different preference weights [23], [24], [25]. Envelop Q-learning [25] 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 approach for tackling multi-objective problems with linear preferences.

IV. MFC-EQ

This section presents the design of our learning framework for decentralized MAiF. We first outline the learning environment, including agents' observation, communication, action, and reward functions. We then elaborate on the bi-objective multi-agent learning process based on mean-field theory and envelope Q-learning.

A. Environment and Model Design

1) Observation: As with most research in the MAPF community [1], we study our problem in a 2D 4-neighbor grid environment. To mimic many real-world robotics applications where robots have limited visibility and sensing range, each agent in our grid world can observe only 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 [9], [10], the third channel encompasses heuristic information, where each cell in the FOV is assigned a value proportional to the short-path distance from that cell to the agent's goal.

2) Action: Agents can move to their cardinally adjacent cells at each time step. The action taken by agent i at time step t, denoted by $a_t^i \in \mathbb{R}^5$, is a 5-dimensional one-hot vector where each dimension represents one of the actions: $\{up, down, left.right, wait\}$. The first four actions move the agent to a new cell, shifting its observation accordingly. The $wait$ action keeps the agent in its current cell, which is especially crucial for formation control, allowing other agents to catch up and reduce formation deviation.

3) Multi-Agent Communication: To maintain the desired formation, agents need to not only communicate with nearby agents within their FOVs but also with those outside. We specifically design communication messages to convey critical information under low communication bandwidth.

In many real-world robotics applications, each agent can only access pairwise relative positions between itself and other agents. 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 $\ell_{\boldsymbol{g}} = \langle \boldsymbol{g}^1, \ldots, \boldsymbol{g}^M \rangle$. The relative position between agent i and j is defined 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 knows the relative positions in the goal formation, $\{\boldsymbol{g}^{i,j}\}_{j \in [M]}$, pre-calculated

before execution. With this information, even without knowing its absolute position, an agent can still calculate the formation deviation. As defined in Eq. (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

$$\mathscr{F}(\ell_{\boldsymbol{p}}, \ell_{\boldsymbol{g}}) = \sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^m - \boldsymbol{g}_n^m - \boldsymbol{C}_n) - \boldsymbol{\Delta}'_n|,$$

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. Substituting $\boldsymbol{C}$ with $\boldsymbol{p}^i - \boldsymbol{g}^i$, we then rewrite the formation deviation using only relative position:

$$\begin{aligned}
&\sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^m - \boldsymbol{g}_n^m) - \boldsymbol{\Delta}_n| \\
=&\sum_{m=1}^{M} \sum_{n=1}^{d} |[(\boldsymbol{p}_n^m - \boldsymbol{p}_n^i) - (\boldsymbol{g}_n^m - \boldsymbol{g}_n^i)] - \boldsymbol{\Delta}_n^*| \\
=&\sum_{m=1}^{M} \sum_{n=1}^{d} |(\boldsymbol{p}_n^{i,m} - \boldsymbol{g}_n^{i,m} - \boldsymbol{\Delta}_n^*|
\end{aligned}$$

where $\boldsymbol{\Delta}_n^*$ 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 based on only the relative positions with time complexity $\mathcal{O}(d \cdot M)$. Additionally, agents can easily infer the mean action based on relative positions.

Importantly, allowing agents to communicate relative positions does not make the problem centralized. Each agent remains unaware of the surrounding environment of agents outside its FOV and cannot predict their observations or next actions based solely on relative positions.

4) 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 a 2-tuple. The first element concerns the makespan. We adopt the individual cost function from DHC [10]: The moving cost of agent i at time step t with action a^i is

$$c_t^i(s^i, a^i) = \begin{cases} -0.075 & \text{collision-free } a^i \\ -0.5 & \text{collision (with obstacles or agents)} \\ 3 & \text{reach goal} \\ 0 & \text{stay on goal} \end{cases}.$$

Collision-free actions, including moves (up, $down$, $left$, or $right$) and $wait$ (on or away from the goal), are slightly penalized to encourage agents to reach their goals quickly. The second element concerns the formation deviation. As defined in Eq. (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 to minimize it by maximizing rewards. Thus, the reward function is:

$$\boldsymbol{r}_t^j(s_t^j, a_t^j) = (c_t^j, -\mathscr{F}_t^j(\ell_t, \ell^g))^\top. \tag{2}$$

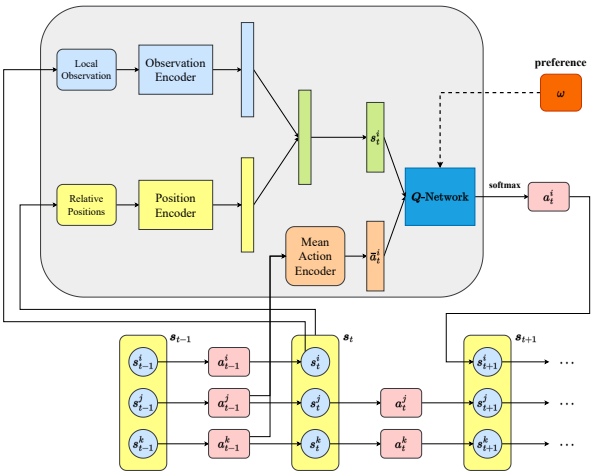

Fig. 2: Illustration of the model architecture of MFC-EQ. The bottom demonstrates the state/observation transition in the partially observable environment. The agent's Q-network gathers information from the environment through partial observation and limited communication and chooses the next action accordingly.

5) Model Architecture: Given the partially observable multi-agent environment, we design a Q-network, with the learning algorithm introduced later. As shown in Fig. 2, the network projects each agent's observations and communication messages into a corresponding action. The local observation and relative positions are fed into two separate encoders: The observation encoder uses stacked convolution layers followed by linear layers, while the relative position encoder uses linear layers. These encodings are then concatenated and passed through another linear layer to obtain the final state representation, s_t^i, for agent i at time step t. The mean action is calculated by collecting other agents' actions from the previous time step. Lastly, stacked linear layers project these inputs to Q-values, conditioning on the state, action, mean action, and given preference. The agent selects its next action that maximizes the Q-function produced by the Q-network.

B. Mean-Field and Envelop Optimality

In the following, we discuss the learning algorithm in detail. Learning multiple policy networks, $\pi = [\pi^1, \ldots, \pi^M]$, for this bi-objective multi-agent task is highly challenging. To simplify this, we make common assumptions.

1) Mean-Field Approximation: The goal of MAiF is to minimize the makespan and formation deviation. With our specifically designed reward function, the return—the discounted sum of all rewards from the initial to the goal joint state, $\sum_t \sum_j \gamma^t \boldsymbol{r}_t^j(\boldsymbol{s}_t, \boldsymbol{a}_t)$—reflects the true values of the two objectives. Therefore, the learning goal is to find a set of policies that maximize the general sum of Q-values $\arg\max_{\pi^1,\ldots,\pi^M} \sum_{j=1}^{M} \boldsymbol{\omega}^\top \boldsymbol{Q}^{\pi^j}(s^j, \boldsymbol{a})$ given the linear preference $\boldsymbol{\omega}$. However, the dimensionality of $\boldsymbol{s}$ and $\boldsymbol{a}$ grows exponentially with the number of agents, rendering efficient learning infeasible. To address this, we introduce

mean-field reinforcement learning. Similar to [15], we first adopt the common assumptions of homogeneity and locality. Homogeneity assumes that each agent shares the same policy, so $\pi^i = \pi^j$ for all $i \neq j$. Locality, derived from partial observability, suggests that agents' actions depend only on their visible surroundings. Assuming actions are represented by one-hot vectors, we define the mean action as:

$$\bar{a}_t^j = \frac{1}{|\mathcal{N}^j|} \sum_{k \in \mathcal{N}^j} a_t^k,\ a_t^k \sim \pi^k(\cdot|s^k, \bar{a}_{t-1}^k), \tag{3}$$

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

$$\begin{aligned}\boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, \boldsymbol{a}_t) &= \frac{1}{|\mathcal{N}^j|} \sum_{k \neq j} \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, a_t^k) \\ &= \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, \bar{a}_t^j),\end{aligned} \tag{4}$$

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 derive the agent's policy function using the softmax parameterization with the Boltzmann parameter β:

$$\pi^j(a_t^j|s_t^j, \bar{a}_{t-1}^j) = \frac{\exp(\beta \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a_t^j, \bar{a}_{t-1}^j))}{\sum_{a \in A^j} \exp(\beta \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t^j, a, \bar{a}_{t-1}^j))}. \tag{5}$$

2) Bellman Optimality Operator: To extend this framework to multi-objective reinforcement learning, we modify the envelop Q-learning [25] by combining the mean-field operator with the envelop optimality operator. We first condition all Q-values on the linear preference $\boldsymbol{\omega}$, as in $\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$. Similar to standard Q-learning [26], the bi-objective multi-agent Bellman optimality operator $\mathcal{T}$ is defined as:

$$\begin{aligned}(\mathcal{T}\boldsymbol{Q})(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\omega}) := \sum_{j=1}^{M} \boldsymbol{r}^j(s_t^j, a_t^j) + \gamma \mathbb{E}_{s_{t+1}} \arg_Q \big\{ \\ \max_{\boldsymbol{\omega}' \in \Omega} \sum_{j=1}^{M} \max_{a^j \in A^j} \boldsymbol{\omega}^\top \boldsymbol{Q}(s_{t+1}^j, a^j, \bar{a}_{t+1}^j, \boldsymbol{\omega}') \big\},\end{aligned} \tag{6}$$

where $\arg_Q$ takes the maximized bi-objective Q-values before the linear scalarization. This operator mirrors the Bellman optimality operator in standard Q-learning for single-agent RL and provides the temporal difference (TD) target. By maximizing $\boldsymbol{\omega}'$ over the next state and its onward trajectory, this approach offers an optimistic perspective on its future rewards. Iteratively applying this operator to the Q-function allows for convergence on a near-optimal Q-function [25].

C. Double Q-learning

Off-policy RL algorithms allow for greater exploration in the state-action space and have been adopted by several learning-based MAPF solvers, such as in [10]. We thus design our learning algorithm based on the double Q-learning [27] with two different loss functions. 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. Once enough transitions are collected in the replay buffer, the learning phase begins (Line 11-14). Given a mini-batch of N transitions and $N_{\boldsymbol{\omega}}$ preferences, the TD target $\boldsymbol{y} = (\mathcal{T}\boldsymbol{Q})(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$ is estimated via Eq. (6). The first loss function is computed as the L_2-norm of the multi-objective TD:

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

Although this loss function closely estimates the true expected return, its non-smooth surface makes the early learning steps challenging. We combine this with an additional loss function using the projected TD:

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

$L_A(\theta)$ provides a closer estimation of the true Q-function, while $L_B(\theta)$ smooths the optimization landscape. We train the Q-network using homotopy optimization [28] based on the combination of these two loss functions:

$$L(\theta) = (1 - \zeta) L_A(\theta) + \zeta L_B(\theta), \tag{7}$$

where, in our case, we gradually increases ζ from 0 to 1 exponentially as learning progresses.

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 = $1, \ldots, E$ **do**
4 Initialize $\bar{a}_0^j$ for all $j \in [M]$
5 **for** $t = 1, \ldots, T_{max}$ **do**
6 Sample $\boldsymbol{\omega} \sim \Omega$ and then a_t^j from Eq. (5)
7 Compute new mean actions $\bar{a}_t^j$ by Eq. (3) for all $j \in [M]$
8 Take the joint action $\mathbf{a}_t = [a_t^1, \ldots, a_t^M]$ from the state $\mathbf{s}$ to the next state $\mathbf{s}_{t+1}$
9 Compute the reward $\mathbf{r}_t = [r^1, \ldots, r^M]$ by Eq. (2)
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 $\bar{\boldsymbol{a}}_t = [\bar{a}_t^1, \ldots, \bar{a}_t^M]$ is the collection of mean actions
11 **if** update **then**
12 Sample N transitions from $\mathcal{D}$ and $N_{\boldsymbol{\omega}}$ preferences from Ω
13 Compute the TD target using the operator in Eq. (6)
14 Update Q_θ by minimizing the loss from Eq. (7)
15 Update $\boldsymbol{Q}_{\bar{\theta}}$ with the learning rate α: $\bar{\theta} \leftarrow \alpha\theta + (1 - \alpha)\bar{\theta}$
16 Increase ζ along the predefined homotopy path

V. Empirical Evaluation

This section presents our experimental results conducted on a 2.3GHz Intel Xeon server with 8 NVIDIA A40 GPUs.

A. Experimental Setups

We use 4-neighbor grids without placing obstacles in the top-left and bottom-right corners. The default obstacle density in the remaining areas is set to 10%. Agents start at the top-left corner and move toward the bottom-right corner. Following existing learning-based MAPF solvers, the FOV size is set to 9×9. The formation in the goal position defines

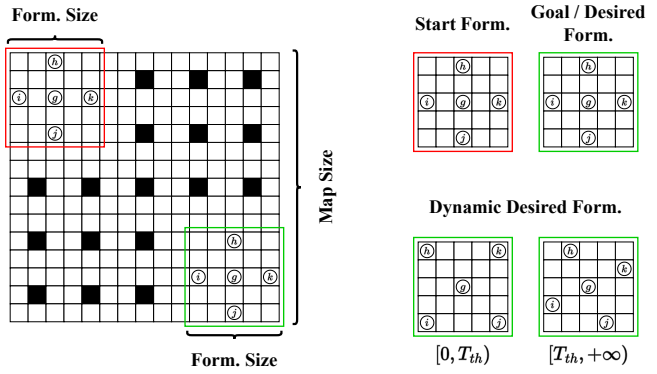

Fig. 3: Demonstration of experiment environments.

the desired formation. For each data point in the results, we average the outcomes over 100 samples, generated from 10 random maps and 10 random formations.

B. Training of MFC-EQ

Similar to [10], we employ the scheme of curriculum learning [29] to gradually introduce more challenging tasks into the training data. For each formation size, the training starts with a 30×30 map and 5 agents, and we will increase either the map size or the number of agents once the success rate exceeds 0.9. The final model was trained for $500,000$ episodes with a batch size of 192.

C. Centralized Baselines

We compare our method against the following centralized planning methods.

1) Scalarized Prioritized Planning (SPP): Since solving MAiF optimally is NP-hard, we developed an efficient yet suboptimal baseline based on the prioritized planning algorithm [30]. Each agent is given a unique priority, and paths are planned sequentially using a low-level A* search that respects the paths of higher-priority agents. The A* search uses a scalarized f-value combining the makespan f-value and the formation deviation f-value. For any node n, we define its cost from the start node to node n as T_n, the scalarized f-value can be written as:

$$f(n) = \lambda\big[T_n + \mathit{dist}(\boldsymbol{v}^i_{T_n}, \boldsymbol{g}^i)\big] + (1-\lambda)\sum_{t=1}^{T_n} \mathscr{F}^i_t(\boldsymbol{v}^i_t, \boldsymbol{g}^i),$$

where $\mathit{dist}(\cdot,\cdot)$ is the Manhattan distance and $\mathscr{F}^i_t$ represents the partial formation deviation involving agent i and higher-priority agents at time step t as defined in Eq. 1. Although this baseline is incomplete, it can often find a solution quickly. However, the quality, especially for formation deviation, may suffer in congested environments with many agents. Unlike SWARM-MAPF, this planner can target any given linear preference. We tested its performance by varying λ from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

2) SWARM-MAPF (SWARM): SWARM-MAPF [7] is a state-of-the-art centralized method that integrates swarm-based formation control with a MAPF algorithm. It operates in two phases: In Phase 1, SWARM-MAPF first calculates the lower bound of the makespan $B = \max_{1\le i\le M} \mathit{dist}(s_i, g_i)$ and then selects a leader to plan a path of length bounded by the user-specified parameter w multiplied by B, ensuring that the path is sufficiently distant from obstacles to allow other agents to preserve formation while avoiding obstacles. In Phase 2, SWARM-MAPF employs a modified conflict-based search [31] to replan critical segments while minimizing the makespan. While SWARM-MAPF is complete, it cannot specifically target a given preference, as the trade-off between objectives cannot be directly controlled through the parameter w.

3) Joint State A (JSA*):* Joint state A* [32] applies the ϵ-constraint search algorithm [33] directly 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 by makespan, while the FOCAL list breaks ties based on formation deviation [7]. Due to the exponential growth of the joint state space with the number of agents, this method is only feasible for small instances (fewer than 5 agents in our setups). Varying ϵ in the focal search guarantees finding the Pareto-optimal frontier.

TABLE I: Results of MFC-EQ with different preferences evaluated by different scalarized 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**

D. Main Results

1) Linear Preferences: We first evaluate the ability of our learned Q-network to adapt to different preferences using an environment with 16 agents and a 9×9 formation size on a 48×48 map. We test different preferences $\boldsymbol{\omega} = (\lambda, 1-\lambda)^\top$ 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 I shows 5 different solutions, with each MIX column highlighting the solution that minimizes the projection onto that particular preference. We observe that each MIX(λ) objective is minimized when the corresponding preference $\boldsymbol{\omega}(\lambda)$ is fed into the Q-network. The results suggest that the Q-network trained with MFC-EQ can adapt to different preferences, producing multiple solutions tailored to the given preferences.

2) Numbers of Agents: We evaluate MFC-EQ with different numbers of agents across different map sizes and compare the results with centralized baselines. For SPP and MFC-EQ, λ is set to 0.5. For SWARM, w is set to 1.0. The runtime limit is 30 seconds for MFC-EQ and 5 minutes for SWARM and SPP. As shown in Table II, MFC-EQ does not always achieve perfect success rates due to the partially observable environment but maintains relatively high and

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

Map Size	M	Success Rate			Makespan			Form. Dev.			MIX(0.5)		
		SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ
32 × 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**
	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 × 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**
	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 × 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**
	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 III: Results for MFC-EQ and centralized baselines with different formation sizes in various sizes of grids.

Map Size	Form Size	Success Rate			Makespan			Form. Dev.			MIX(0.5)		
		SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ	SPP	SWA-RM	MFC-EQ
32 × 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
	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 × 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**
	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 × 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**
	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**

acceptable rates. It generally achieves slightly lower success rates than SWARM for large numbers of agents, despite having a much shorter runtime limit. MFC-EQ often outperforms SWARM when evaluated against the given preference. These results suggest that MFC-EQ scales well to large numbers of agents across different map sizes.

3) Formation Sizes: We repeat the above experiment with different formation sizes, using various obstacle-free corner sizes and randomly generated desired formations. The larger the corner, the more spread out the formation tends to be. The number of agents is fixed at 16. As shown in Table III, smaller formations are generally more challenging, resulting in larger makespans and formation deviations. Compared to SWARM, SPP typically achieves a better makespan but much worse formation deviation. MFC-EQ outperforms the two baselines with respect to both objectives in most cases.

4) Dynamic Formation: We tested these methods for more challenging tests where agents were required to adjust to different formations on the fly. Specifically, the desired formation changes to a different one at $T_{th} = 30$. Centralized methods cannot handle such tasks effectively, as agents' paths must be planned before execution. In contrast, MFC-EQ allows each decentralized agent to be notified of the new formation, resulting in updated calculations of relative positions, which enables the agents to seamlessly adjust to the new formation. The results shown in Table IV suggest

TABLE IV: Results for MFC-EQ and centralized baselines for tests with a dynamic formation.

M	Success Rate			Makespan			Form. Dev.			MIX(0.5)		
	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**

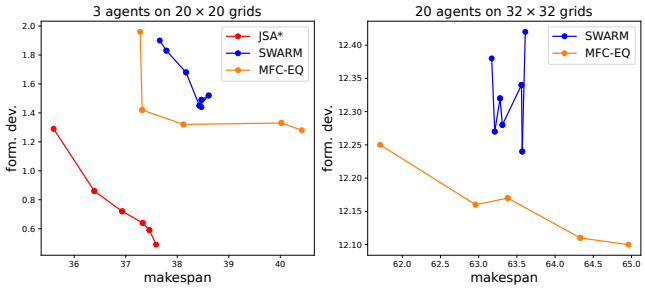

Fig. 4: Trade-off of makespan and formation deviation.

that MFC-EQ offers the flexibility needed to manage changing formations, whereas other methods result in significantly larger formation deviations.

5) Makespan and Formation Trade-off: We further compare MFC-EQ with other methods across different preferences. Due to the limited scalability of JSA*, we first use instances with a 20×20 map size, 3×3 formation size, 15% obstacle density, and 3 agents. We vary ϵ from 1.0 to 1.8 for JSA*, w from 1.0 to 1.6 for SWARM, and λ (for $\boldsymbol{\omega}$) from 0.1 to 0.9 for MFC-EQ. JSA* can provide the Pareto-optimal frontier only for small-scale instances. We then repeat this experiment on larger instances with a 32×32 map size, 9×9 formation size, and 20 agents. The results are shown in Fig. 4. SPP produces solutions with near-optimal makespans but significantly larger formation deviations compared to the other methods. In large-scale cases, SWARM tends to fluctuate, meaning that even with more makespan allowance, it may result in worse formation deviations. MFC-EQ generates a near-convex envelope that encompasses all solutions from SWARM, although it remains suboptimal. Additionally, it offers a wider range of makespan options with greater solution variety.

VI. Conclusion and Future Work

We proposed MFC-EQ, a general Q-learning framework for solving decentralized MAiF under conditions of partial observation and limited communication. MFC-EQ leverages mean-field approximation to simplify complex multi-agent interactions and employs envelop Q-learning to adapt to various preferences in this bi-objective task. Our theoretical analysis confirms that the combination of these two operators can still converge to a fixed optimum. Empirical results

demonstrate that MFC-EQ outperforms existing centralized baselines in most scenarios and is particularly versatile and effective in handling dynamically changing desired formations. Moreover, MFC-EQ is not restricted to solving MAiF and has significant potential for generalization to other multi-objective tasks in multi-agent systems.

While MFC-EQ shows strong performance in this initial attempt, there is certainly room for improvement, as indicated by the experimental results. One promising direction is to design a mixing network [34], [35] to estimate the joint action-value function for handling both objectives. Another direction is to apply other multi-objective RL algorithms that do not request the setting of linear preference, such as [36], [21], [37]. Regarding the agents' policy networks, integrating the current architecture with more advanced network designs [38] and enhanced communication mechanisms [39], [40], [41] might further improve performance. Last but not least, we could also incorporate high-level global guidance as in [9], [42] for both objectives. We leave these directions for future exploration.

REFERENCES

[1] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *SoCS*, 2019, pp. 151–158.

[2] H. Ma and S. Koenig, "Ai buzzwords explained: multi-agent path finding (mapf)," *AI Matters*, vol. 3, no. 3, pp. 15–19, 2017.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[4] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, and S. Koenig, "Planning, scheduling and monitoring for airport surface operations." in *AAAI Workshop: Planning for Hybrid Systems*, 2016, pp. 608–614.

[5] H. Ma, J. Yang, L. Cohen, T. Kumar, and S. Koenig, "Feasibility study: Moving non-homogeneous teams in congested video game environments," in *AIIDE*, 2017, pp. 270–272.

[6] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *ICIIS*, 2012, pp. 1–5.

[7] J. Li, K. Sun, H. Ma, A. Felner, T. Kumar, and S. Koenig, "Moving agents in formation in congested environments," in *SoCS*, 2020, pp. 131–132.

[8] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[9] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *IROS*, 2020, pp. 11 748–11 754.

[10] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *ICRA*, 2021, pp. 8699–8705.

[11] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings*. Elsevier, 1994, pp. 157–163.

[12] Q. Lin and H. Ma, "Sacha: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 2377–3766, 2023.

[13] S. Liu, L. Wen, J. Cui, X. Yang, J. Cao, and Y. Liu, "Moving forward in formation: a decentralized hierarchical learning approach to multi-agent moving together," in *IROS*, 2021, pp. 4777–4784.

[14] H. E. Stanley, *Phase transitions and critical phenomena*. Clarendon Press, Oxford, 1971, vol. 7.

[15] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *ICML*, 2018, pp. 5571–5580.

[16] S. G. Subramanian, M. E. Taylor, M. Crowley, and P. Poupart, "Partially observable mean field reinforcement learning," in *AAMAS*. IFAAMAS, 2021.

[17] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning." in *ICML*, 1998, pp. 197–205.

[18] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning," *NeurIPS*, 2001.

[19] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *ICML*, 2005, pp. 601–608.

[20] S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli, "Policy gradient approaches for multi-objective sequential decision making," in *IJCNN*, 2014, pp. 2323–2330.

[21] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[22] X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning," in *IROS*, 2019, pp. 977–983.

[23] A. Castelletti, F. Pianosi, and M. Restelli, "Multi-objective fitted q-iteration: Pareto frontier approximation in one single run," in *ICNSC*, 2011, pp. 260–265.

[24] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *ICML*, 2019, pp. 11–20.

[25] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *NeurIPS*, 2019.

[26] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[27] H. Hasselt, "Double q-learning," *NeurIPS*, 2010.

[28] L. T. Watson and R. T. Haftka, "Modern homotopy methods in optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 74, no. 3, pp. 289–305, 1989.

[29] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[30] D. Silver, "Cooperative pathfinding," in *AIIDE*, 2005, pp. 117–122.

[31] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[32] J. Pearl and J. H. Kim, "Studies in semi-admissible heuristics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 4, pp. 392–399, 1982.

[33] Y. Haimes, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 296–297, 1971.

[34] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[35] T. Hu, B. Luo, C. Yang, and T. Huang, "Mo-mix: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[36] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *ADPRL*. IEEE, 2013, pp. 191–199.

[37] S. Mannor and N. Shimkin, "A geometric approach to multi-criterion reinforcement learning," *JMLR*, vol. 5, pp. 325–360, 2004.

[38] C. He, T. Yang, T. Duhan, Y. Wang, and G. Sartoretti, "Alpha: Attention-based long-horizon pathfinding in highly-structured areas," in *ICRA*. IEEE, 2024, pp. 14 576–14 582.

[39] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1455–1462, 2021.

[40] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang, "Multi-agent path finding with prioritized communication learning," in *ICRA*. IEEE, 2022, pp. 10 695–10 701.

[41] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding," in *IROS*, 2023, pp. 9301–9308.

[42] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.