Curriculum Learning Based Multi-Agent Path Finding for Complex Environments

Cheng Zhao, Liansheng Zhuang*, Yihong Huang, Haonan Liu
University of Science and Technology of China, Hefei, China
Email:{zc16@mail.ustc.edu.cn, lszhuang@ustc.edu.cn}

2023 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-8867-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCNN54540.2023.10191932

***Abstract*—Multi-agent reinforcement learning (MARL) is a promising tool to solve the Multi-Agent Path Finding (MAPF) task, which aims to find conflict-free paths for multiple agents, one for each agent, from a start position to a goal position. It uses global information to learn a mechanism for cooperation among agents by maximising the cumulative team rewards, which are often very sparse. However, the sparsity of rewards implies that agents have to blindly explore all possible paths, which makes MARL methods difficult to converge in complex environments. To address this issue, this paper proposes a novel Curriculum based Path-finding Learning (CPL) under the framework of curriculum learning, which allows agents to start with simple skills and to learn cooperative strategies stage-by-stage for more efficient training. Specifically, CPL divides the training process into three stages and speeds up the learning by changing the difficulty of the tasks from easy to hard. Experiments on random obstacle grid worlds show that our proposed method performs significantly better in terms of success rate and makespan than state-of-the-art learning-based methods.**

***Index Terms*—Multi-agent path finding; Multi-agent reinforcement learning; Curriculum learning**

I. Introduction

Multi-agent path finding (MAPF) is an NP-hard problem [1] that describes how a group of agents move from their start positions to goal positions in an environment with obstacles. It arises in many real-world applications of multi-agent systems, such as warehouse robots [2], office robots [3], aircraft-towing vehicles [4], etc. The objective of the MAPF task is to plan a set of conflict-free paths, one for each agent, and to minimise a global cost function. The common cost functions include *sum-of-cost* (i.e. the sum of the number of time steps for all agents to reach their destinations.) or *makespan* (i.e. the time until all agents have reached their destinations). A major challenge of MAPF is to avoid frequent collisions and blockages, as multiple agents interact with each other. Many search-based methods perform well in finding collision-free solutions using global information in simple environments, but poorly in terms of real-time performance and scalability. Specifically, the classical planners have to re-plan all paths even when the scene changes slightly, and the computational complexity of the methods is exponential to the number of agents. It is intractable to apply these classical methods in real-time and large-scale tasks.

*This work was supported in part to Dr. Liansheng Zhuang by NSFC under contract No.U20B2070 and No.61976199. Dr. Liansheng Zhuang is the corresponding author.

Recently, a great amount of work focus on learning-based methods for better real-time performance and scalability. Learning-based methods generally model a MAPF task as a decentralized partially observable Markov decision process (Dec-POMDP) [5] and use reinforcement learning to solve this sequential decision problem. Unlike search-based methods that require global information to compute complete paths for all agents, learning-based methods only require local information and allow each agent to plan the optimal one-step path given its current observation. However, because each agent's observation is localised, distributed learning-based approaches usually result in poor cooperation among agents [6]. For the purpose of achieving cooperation, existing methods try to utilize extra (or global) information either in their online planning phase or in their training phase. In the former case, agents receive information about the observations from other agents by means of communication during the execution [7]–[9]. But in practice, such methods still suffer from communication delays and communication jamming. The other case is to use the global information during the training to learn a mechanism for cooperation, while the individual local observations of agents are used for decision making during the execution [10]–[12]. Multi-agent reinforcement learning (MARL) is a representative approach in this case. The centralized training with decentralized execution (CTDE) paradigm, which uses global information during the training while allowing agents to make decisions independently during the execution, is gaining more and more attention [13], [14]. In general, such approaches have more independent decision-making processes and in practice offer better robustness than the former approaches.

Though having achieved excellent performance, the CTDE methods still have some problems in solving MAPF tasks. In MAPF, however, valuable team rewards are always very sparse. For example, all agents receive a positive team reward only when the entire MAPF task is completed, i.e. when all agents avoid collisions and reach their destinations. This results in a training process where all agents have to blindly explore all possible paths before completing the task once. Moreover, in a multi-agent environment, especially a complex one, the probability of all the agents reaching their respective destinations in a blind exploration is very low, making training difficult to converge. In fact, unlike other cooperative multi-agent environments, MAPF can be explicitly broken down into several sub-tasks, each performed by a single agent. Therefore,

the common dense individual rewards can be used to guide agents through individual sub-tasks, getting a feasible strategy more quickly for completing the team task. Unfortunately, existing methods based on CTDE often ignore the dense individual rewards. As a result, typical CTDE methods usually perform poorly for complex MAPF tasks.

Motivated by the above insights, this paper proposes a novel Curriculum based Path-finding Learning (CPL) for MAPF tasks under the CTDE paradigm. In order to have better learning efficiency in complex environments, our CPL is based on the curriculum learning framework [15] in which agents start with simple single-agent path-finding skills and then progressively learn cooperative strategies by network parameter inheritance. To this end, CPL divides the training process into three stages to perform the learning tasks from easy to difficult, just like a human. The first stage motivates an agent to complete single-agent tasks by individual rewards. The second stage motivates agents to complete their respective path-finding tasks in a multi-agent environment still by individual rewards. The third stage motivates agents to cooperate with each other and complete the entire team task by team rewards. Although the team rewards here remain sparse, the first two stages allow agents to quickly find paths with high values, thus resolving the difficulties in exploration. In addition, CPL has good generalisation, as it trains on randomly generated training maps and performs well on test maps that have never been seen before.

Our main contributions can be summarized as follows:

- We propose a novel CPL framework for the MAPF task under the CTDE paradigm to learn a cooperation mechanism of agents and to guarantee the advantages of distributed decision-making.
- We propose to counter the exploration problem posed by sparse team rewards via curriculum learning, thus improving the efficiency of training.
- We conduct experiments on 20*20 random obstacle grid worlds with different obstacle densities and different numbers of agents, to show that CPL outperforms state-of-the-art learning-based methods in terms of success rate and makespan, especially in crowded environments.

II. Related Work

Over the last few decades, many methods have been proposed to solve the MAPF tasks, which can be divided into two categories, search-based methods and learning-based methods.

A. Search-based methods

Search-based methods generally search in a specific state-space to find an optimal or bounded sub-optimal solution. A common straight-forward derived state-space is denoted as the *k-agent* state-space where the states are the different ways to place k agents at different positions in the map, one agent per position. Operators between states are all non-conflicting combinations of actions (including wait) that can be taken by the agents. A* is a general-purpose algorithm that is well suited to searching *k-agents* state-space to solve MAPF. A* with an admissible heuristic is proved as an optimal solver, where a simple admissible heuristic is to sum the individual heuristics of the single agents such as Manhattan distance. However, A* for MAPF has a major drawback in that the size of the state-space is exponential in the number of agents. A prominent example of A*-based sub-optimal algorithm is Hierarchical Cooperative A* (HCA*) [16]. In HCA* the paths of agents are planned one at a time according to some predefined order. Once a path to the goal is found for the first agent, that path is written into a global reservation table. Besides, M* [17] and its enhanced variant ODrM* [18] are also A*-based algorithms. M* dynamically changes the dimensionality and the branching factor based on conflicts. The dimensionality is the number of agents that are not allowed to conflict.

Another optimal MAPF solver not based on *k-agent* state-space is Conflict Based Search (CBS) [19]. CBS consists of two parts: the high-level of CBS searches the constraint tree (CT), where each CT node contains a set of constraints imposed on the agents derived from conflicts occurring in ancestral nodes, and a single solution consistent with these constraints. The search process selects a CT node with the lowest cost of the solution, and then verifies whether there are still collisions among agents. If so, a new CT child node is generated and a new constraint is added to the child node. Then, the low-level of CBS is where the optimal path is solved for each agent independently, given the constraints of a CT node, using a basic search method like A*. Moreover, many variants derived from CBS are still popular today. Enhanced CBS (ECBS) [20] introduces the focal A* method to select CT nodes by always choosing the node that is currently closest to the target among all nodes to be expanded with suboptimality guarantee. Explicit Estimation CBS (EECBS) [21] maintains a value function using online learning to filter the fraction of all CT nodes that still have high value and satisfy the sub-optimal conditions. Flexible EECBS (FEECBS) [22] follows the way the high-level of EECBS searches for CT nodes and improves on the low-level solves for paths subject to constraints and bounded suboptimality.

Search-based methods tend to have sophisticated theoretical guarantees, so they return high-quality solutions. However, the search space of these methods grows exponentially with the number of agents, so these methods are not suitable for large-scale situations. In addition, search-based methods solve each task independently with no reusable parts. In other words, even if the scene changes slightly, for example by moving the start position of an agent in the map, the entire task must be re-solved. This is a fatal problem in practice, as real-world scenarios always change very frequently.

B. Learning-based methods

Recently, more and more work applies deep learning, particularly deep reinforcement learning, to MAPF tasks. Learning-based methods generally model the MAPF task as a Dec-POMDP [5], letting agents learn with a well-designed reward function to reach destinations without any collisions. To enable cooperation among agents, one approach is to introduce local

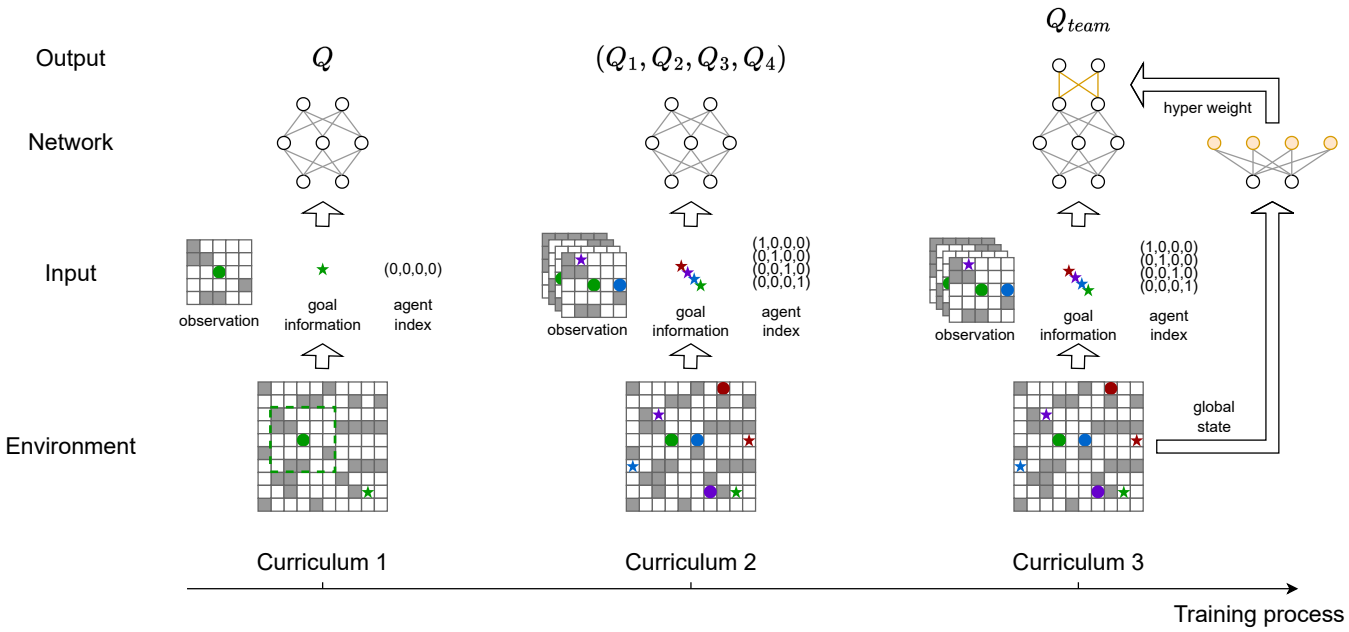


Fig. 1. The training process for CPL.

communication. Agents are informed of the personal information of surrounding agents to facilitate long-horizon path-finding and cooperation. DHC [7] formalizes the multi-agent environment as a graph and allows agents to communicate with neighbors via graph convolution. It treats each agent independently and leverages single-agent Q-learning for multi-agent partially observable Markov game. PICO [8] includes two phases, implicit priority learning phase and prioritized communication learning phase. In the implicit priority learning, PICO aims to build an assisted imitation learning task that predicts the local priority of each agent by imitating classical coupled planner (e.g., ODrM*). In the prioritized communication learning, the obtained local priorities are used to generate a time-varying communication topology consisting of agent clusters, where these priorities are considered as the weights of an ad-hoc routing protocol. These communication-based approaches are indeed intuitive, but in practice they also suffer from communication delays, communication jamming and other problems.

Another approach is to use global information to associate all the agents together for cooperation during the training. PRIMAL [10] combines reinforcement learning and imitation learning to learn fully distributed policies for each agent. PRIMAL chooses the centralized planner ODrM* as the expert that coordinates all agents during the imitation learning episode, whose behavior the agents learn to imitate, allowing them to learn coordinated behaviors, and randomly selects at the beginning of each episode whether it will involve reinforcement learning or imitation learning. MATS [11] employs the multi-step ahead tree-search strategy in single-agent reinforcement learning and imitation learning scheme to fit the results of the tree search strategy to solve the MAPF problem. ME-MADDPG [12] adopts the framework of multi-agent deep deterministic policy gradient to directly map partially observed information to motion commands for multiple agents, and introduces a strategy named mixed experience to train more brilliant agents that can adapt to more complex environments.

A key point of learning-based methods is that the decisions of agents are decentralised, i.e. each agent has its own computing power and plans its own path by receiving only local observations. This not only makes the computational complexity not to grow exponentially with the number of agents, but also avoids the problem of having to re-plan when changes occur. However, with a distributed premise, it is difficult to tell whether an agent's decision is selfish or cooperative. A completely selfish agent considers only its own goal and ignores the behaviour of other agents, often leading to a large number of collisions and blockages. While a completely cooperative agent benefits from avoiding collisions with other agents, which may lead to losing its own goal. Therefore, how to learn collaborative strategies efficiently without losing individual goals is an important topic in solving MAPF with learning-based methods.

III. Methodology

A. Overview

As shown in Fig.1, our CPL consists of three curriculums, which are arranged from easy to difficult. The coloured squares in the figure represent the agents and the coloured stars represent the goal positions corresponding to the agents of the same colour. In each curriculum, the agents take the network inputs from the environment, output the values of the

actions in the current state via the Q-value network, and then generate an ϵ-greedy policy for selecting the next action based on that values. Each curriculum lasts several episodes as the training process progresses. At the end of each curriculum, the parameters of the current Q-value network are retained and the curriculum moves on to the next one. Under the CPL framework, agents start with simple path-finding skills and gradually learn to assess team values to generate cooperative strategies.

Note here that, in CPL, each curriculum is under the framework of reinforcement learning. The necessary components of reinforcement learning, such as the state embedding and the reward function, are described in sections B and C, as all curriculums use these components in the same way. Besides, the differences between the curriculums, in the data generated by the environment and the way the gradients are updated, are described in detail in section D.

B. Input embedding

The input to an agent's Q-value network, denoted as observation state o, can be divided into three parts. First, we consider the MAPF task in a partially-observable discrete grid world like that mentioned in PRIMAL [10]. It is difficult for agents to know information about the entire grid world in practice, so they make decisions based on information observed in a fixed-size window centered on themselves. To allow agents to efficiently extract the observation in the finite window, we decompose it into four channels, each of which is a two-dimensional matrix of the same size as the window. The first channel is a binary matrix that identifies obstacles. The second channel marks where other agents are located in the window. The third channel marks the goal location of other agents in the second channel, used to predict the moving direction of the neighbors. The fourth channel marks the agent's own goal location. If the above goals are not in the window, the projection of the position in the window would be given.

Second, in addition to the observation in the finite window, an agent also needs the information about its goal. For this purpose, we include in the input a vector and a scalar, representing the direction of its goal and the Euclidean distance to its goal, respectively.

Third, we also include the index of an agent as part of the input. This is because we allow multiple agents to share the same Q-value network parameter, but make different agents to behave differently even with the same Q-value network. We input each agent's index as a one-hot vector to the network, as shown in curriculum 2 and curriculum 3 in Fig.1. For single-agent case in curriculum 1, the index part of the input is filled with zeros to keep the shape of the input tensor.

C. Reward function

MAPF environments involve multiple sub-tasks, each performed by a single agent. Existing learning-based methods typically motivate agents to complete each sub-task by individual rewards [8], [10], [23]. However, such a reward function being simply used may ignore cooperation among agents. Therefore, path-finding tasks defined in a multi-agent setting require a well defined team reward function [13], [14] in order to incentivise cooperative policies by rewarding good performance on individual sub-tasks, while penalising non-cooperative greedy policies that may benefit individual agents but fail the global objective.

The team reward, given by $R^{\text{te}}(\mathbf{o}, \mathbf{a})$, where $\mathbf{o}$ is the joint observation state and $\mathbf{a}$ is the joint action, is usually associated with the overall task-related joint performance of all agents. We ultimately accomplish the entire MAPF task by maximising cumulative team rewards. We set our team reward function according to the reward function in PRIMAL [10]. At a certain time step, the team reward received by the whole team is equal to the sum of the rewards received by each of the agents in PRIMAL. That is, each agent's action will count as part of the team reward for that time step. An agent that makes a general move counts a reward of -0.3, an agent that stay still off the goal counts a reward of -0.5, and an agent that collides with other agents counts a reward of -2. When the entire MAPF task is completed, the team will receive a team reward of value $+20 * k$, where k is the number of agents.

Notice that such team rewards are actually sparse that the team is only rewarded when the entire task is completed. This leads to difficulties in exploration as the reinforcement learning method struggles to find a high-value trajectory during the training. In this work, we also remain individual rewards to help with the exploration in the first two curriculums. The individual rewards, given by $\mathbf{R}^{\text{in}}(\mathbf{o}, \mathbf{a}) = (R_1^{\text{in}}(o_1, a_1), R_2^{\text{in}}(o_2, a_2), \ldots, R_k^{\text{in}}(o_k, a_k))$, are independently received by each agent at each time step. In order to make the individual rewards work to guide exploration, we have slightly modified the rewards in PRIMAL to make them more dense. Instead of receiving a reward of +20 when completing the entire MAPF task, an agent receives a reward of +20 earlier when it completes its own goal. Our final individual rewards for each agent are shown in Table I.

TABLE I
THE INDIVIDUAL REWARD.

Individual Action	Reward
Move to the goal at the first time	+20.0
Agent collision	-2.0
Common move	-0.3
Stay off the goal	-0.5
Stay on the goal	0.0

D. Curriculum Learning

The optimization goal of multi-agent reinforcement learning is to maximize the cumulative team rewards. However, the team rewards are quite sparse in the MAPF task, as defined in Section C. An appropriate approach is to utilise dense individual rewards to assist with training. The curriculums in CPL are set up to exploit individual rewards, allowing agents to learn cooperative team strategies stage by stage, starting from simple individual values.

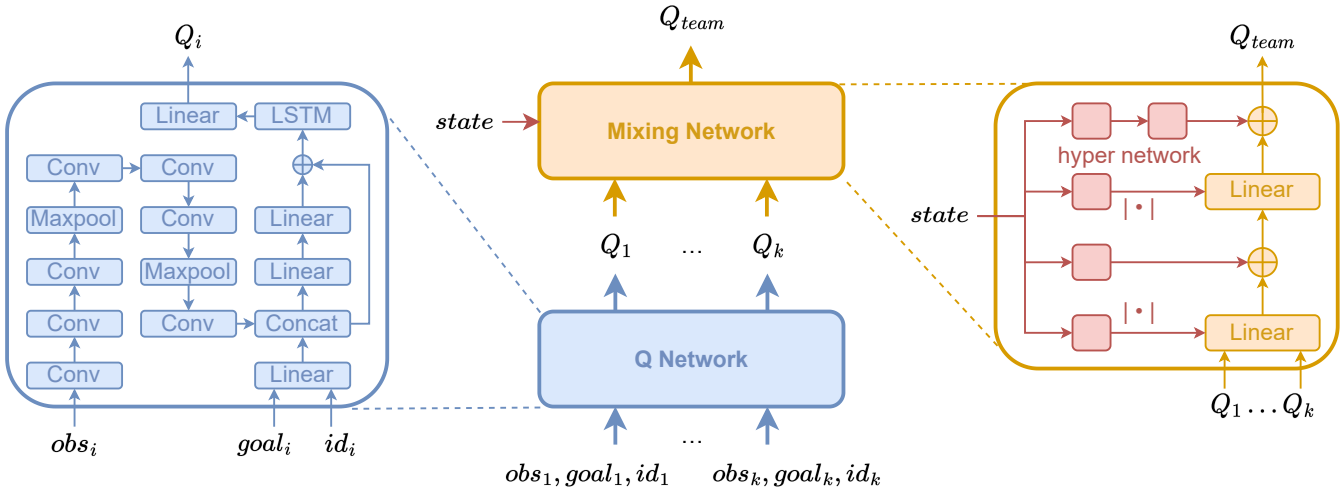


Fig. 2. The network structure of CPL.

a) Curriculum 1 - Single-Agent Path Finding: In the first stage, all training data is generated by a single agent in obstacle grid worlds. The agent feeds the observation, the goal information and the agent index (all zero) to a deep Q-value network (DQN) [24] which outputs action values. The network structure is the same as the agent network used in PRIMAL [10], consisting of linear layers, convolutional layers and an LSTM layer, as shown in the left part of Fig.2. Then, we calculate the TD error between the individual reward and the network output as a loss function for network training:

$$L_1 = [(R^{\text{in}} + \gamma \max_{a^{t+1}} \tilde{Q}(o^{t+1}, a^{t+1}|\theta)) - \tilde{Q}(o^t, a^t|\theta)]^2. \quad (1)$$

R^{in} is the individual reward received at the current time step. γ is the discount factor. $\tilde{Q}(o^t, a^t|\theta)$ is the estimated action value output by the network given the network parameter θ, and the superscript t denotes the time step. Curriculum 1 is easy to learn and is designed to teach the agent the basic skill of moving towards the destination while avoiding static obstacles.

b) Curriculum 2 - Multi-Agent Path Finding: In the second stage, instead of using the training data generated by a single agent, we directly place k agents required for the application in the grid worlds, generating path-finding data where agents may collide with each other. To be able to exploit the skills learned in curriculum 1, the Q-value network parameter θ for curriculum 2 is inherited directly from the Q-value network at the end of training in curriculum 1. In this stage, we still do not consider the task of cooperation among agents. We use the subscript i on the individual-related variables such as reward, observation state and action to denote the agent index, and the loss function for this stage is:

$$L_2 = \sum_{i=1}^{k} [(R_i^{\text{in}} + \gamma \max_{a_i^{t+1}} \tilde{Q}_i(o_i^{t+1}, a_i^{t+1}|\theta)) - \tilde{Q}_i(o_i^t, a_i^t|\theta)]^2. \quad (2)$$

In effect, each term of the summation is a TD error made by each agent based on the its network output Q-value and its received individual reward. Curriculum 2 focuses on the skills of path-finding and collision avoidance in a multi-agent environment. At this point, each agent's strategy remains selfish, as they only act to achieve their own goal for the reward. However, with dense individual rewards, the agents discover those trajectories that accomplish their goals quickly, alleviating the exploration problem that sparse rewards would pose, in preparation for the next stage.

c) Curriculum 3 - Cooperative Multi-Agent Path Finding: In the third stage, we use the data generated by the same environment as in the second stage. So this stage inherits not only the network parameters learned in curriculum 2, but also the replay buffer in curriculum 2. In this stage, our goal is to evaluate the value of the agents' joint actions to the team $Q_{\text{team}}(\mathbf{o}, \mathbf{a})$ and to select the strategy that makes the most team value. Q_{team} is related to Q_i by $Q_{\text{team}} = \sum_{i=1}^{k} w_i Q_i$. We use a mixing network to fit the weights w_i in the formula. In addition, we make our method able to extract decentralised policies for distributed execution. For consistency, it is necessary to ensure that the global argmax operation performed on Q_{team} yields the same result as a set of argmax operations performed on each Q_i:

$$\underset{\mathbf{a}}{\text{argmax}} Q_{\text{team}}(\mathbf{o}, \mathbf{a}) = \begin{pmatrix} \text{argmax}_{a_1} Q_1(o_1, a_1) \\ \vdots \\ \text{argmax}_{a_k} Q_k(o_k, a_k) \end{pmatrix}. \quad (3)$$

This allows each agent to participate in distributed execution by selecting greedy actions only for its Q_i. Such monotonicity can be enforced by a constraint on the relationship between Q_{team} and each Q_i:

$$\frac{\partial Q_{\text{team}}}{\partial Q_i} \geq 0, \forall i \in \{1, \dots, k\}. \quad (4)$$

TABLE II
THE COMPARISON OF ALGORITHMS IN DIFFERENT ENVIRONMENTS

Methods	8 Agent (0%, 10%, 20%, 30% Obstacle Densities)																							
	CA↓				CO↓				SR↑				MS↓				CR↓				TM↓			
CPL	**0.3**	**0.6**	**1.2**	3.2	**0.0**	**0.0**	**0.0**	**0.0**	**100**	**99**	**94**	**65**	**25**	**31**	**45**	**124**	**0.02**	0.02	0.03	0.03	**111**	**121**	**150**	**282**
PRIMAL	1.9	3.0	3.0	6.0	**0.0**	**0.0**	**0.0**	**0.0**	93	90	48	15	35	63	149	234	0.06	0.05	0.02	0.03	221	233	345	565
PICO	0.6	**0.6**	1.3	**2.3**	**0.0**	**0.0**	**0.0**	**0.0**	100	96	55	25	27	42	135	205	**0.02**	**0.01**	**0.01**	**0.01**	124	143	290	463
Methods	**16 Agent (0%, 10%, 20%, 30% Obstacle Densities)**																							
	CA↓				**CO↓**				**SR↑**				**MS↓**				**CR↓**				**TM↓**			
CPL	**2.8**	**3.7**	5.3	16.1	**0.0**	**0.0**	**0.0**	**0.0**	**100**	**95**	**81**	**22**	**27**	**41**	**84**	**213**	**0.10**	0.09	0.06	0.08	**221**	**249**	**374**	**780**
PRIMAL	6.6	8.3	11.6	17.6	**0.0**	**0.0**	0.1	0.1	92	88	50	3	57	72	176	249	0.11	0.12	0.07	0.07	482	510	766	1396
PICO	3.0	3.9	**5.0**	**8.0**	**0.0**	**0.0**	**0.0**	**0.0**	**100**	**95**	57	7	31	49	145	240	**0.10**	**0.08**	**0.03**	**0.03**	251	299	526	1292
Methods	**32 Agent (0%, 10%, 20%, 30% Obstacle Densities)**																							
	CA↓				**CO↓**				**SR↑**				**MS↓**				**CR↓**				**TM↓**			
CPL	**11.9**	**17.4**	**30.3**	**45.6**	**0.0**	**0.0**	**0.0**	**0.0**	**100**	**92**	**50**	0	**32**	**58**	**159**	256	**0.38**	0.30	0.19	**0.18**	**471**	**564**	**1032**	3603
PRIMAL	26.2	30.5	47.3	98.3	**0.0**	0.4	1.6	2.1	92	72	9	0	54	108	245	256	0.49	0.28	0.19	0.38	958	1094	2227	3431
PICO	14.8	20.6	36.3	83.4	**0.0**	0.2	1.3	1.6	**100**	75	19	0	38	97	225	256	0.39	**0.21**	**0.16**	0.33	551	774	1713	**3176**
Methods	**64 Agent (0%, 10%, 20%, 30% Obstacle Densities)**																							
	CA↓				**CO↓**				**SR↑**				**MS↓**				**CR↓**				**TM↓**			
CPL	**84**	**109**	**101**	**109**	**0.0**	**0.0**	**0.0**	**0.0**	80	**20**	0	0	**92**	**218**	256	256	**0.91**	**0.50**	**0.39**	**0.43**	**1230**	**2204**	7630	8566
PRIMAL	116	171	342	635	0.1	2.3	8.0	36.1	75	7	0	0	111	242	256	256	1.04	0.71	1.34	2.48	2419	3680	6611	9157
PICO	91	128	280	591	0.4	8.8	38.4	130.3	**83**	13	0	0	94	225	256	256	0.96	0.57	1.09	2.31	1473	2621	**5342**	**7714**

We use an architecture consisting of an agent network, a mixing network and a set of hypernetworks [25] to represent Q_{team}. Fig.2 illustrates the overall setup. All agents share a single agent network with the same network structure as in the previous two curriculums. The current individual observation state o_i is input at each time step and the individual action values Q_i is output. The mixing network is a feed-forward neural network that takes the output of the agent network as input and mixes it monotonically to produce the value of Q_{team}. To enforce the monotonicity constraint of (4), the weights of the mixing network are restricted to non-negative values. The weights of the mixing network are generated by separate hypernetworks. Each hypernetwork takes the global state s as input to generate a layer of weights for the mixing network, where the global state s is a multi-channel two-dimensional matrix describing the global grid world. Each hypernetwork includes a series of the same convolution and pooling layers as the agent network, and a linear layer followed by an absolute activation function, to ensure that the weights of the mixing network are non-negative. The output of the hypernetwork is then a vector which is reshaped into a matrix of appropriate size. The bias is generated in the same way, but is not restricted to non-negative values. The final bias is generated by a 2-layer hypernetwork with ReLU nonlinearity. The loss function for the approximator is:

$$\begin{aligned} L_3 =&[(R^{\text{te}} + \gamma \max_{\mathbf{a}^{t+1}} \tilde{Q}_{\text{team}}(\mathbf{o}^{t+1}, \mathbf{a}^{t+1}, s^{t+1}|\theta, \phi)) \\ &- \tilde{Q}_{\text{team}}(\mathbf{o}^{t}, \mathbf{a}^{t}, s^{t}|\theta, \phi)]^2, \end{aligned} \tag{5}$$

where ϕ is the parameter of the hypernetworks. Curriculum 3 makes use of the global state to calculate the TD error between the team reward and the team value, ultimately maximising the cumulative team rewards. Due to Equation (3), the agents can select the actions that maximise the individual values in a distributed manner. This is a paradigm of centralized training with decentralized execution.

IV. EXPERIMENT

A. Experiment Settings

In this section, we conduct experiments in typical random obstacle grid worlds with reference to the setup in PICO [8]. Importantly, to emphasise the generalisation of our methods, all training and test grid worlds are generated randomly, rather than doing reinforcement learning on a specific map. In the training phase, the size of the grid world is fixed at 20×20, and the size of the observation window is fixed at 11×11. The content of the world for each epoch is random. Specifically, the obstacles are randomly generated in the world according to a certain probability (or density), and the start position and the goal position of each agent are randomly generated in a random piece of connected area in the world. To measure the performance of the methods under different settings, the number of agents varies in four cases: 8, 16, 32, and 64, and the density of obstacles varies in four cases: 0%, 10%, 20%, and 30%. In the test phase, 100 different grid worlds are randomly generated with the same number of agents and obstacle density as in the training phase. The final comparison with baselines is based on agents' average performance across all these test grid worlds.

In this work, representatives of learning-based methods, PRIMAL and PICO, are chosen as baselines. PRIMAL [10] is a centralized training with decentralized execution method that uses search-based method ODrM* [18] as a global expert planner to guide agents in imitation learning, combined with reinforcement learning. PICO [8] is a communication-based method, which learns to predict the local priority of each agent by imitation learning and generates a time-varying communication topology based on the obtained local priorities. For all of these methods, including ours, the maximum length of individual paths planned by agents is limited to 256. The performance is evaluated on 6 different measurements as follows:

- Collision with agents (CA) - Number of collisions with other agents.
- Collision with obstacles (CO) - Number of collisions with static obstacles.
- Success rate (SR) - Rate of successful solutions for the entire team.
- Makespan (MS) - The time span to complete the entire task.
- Collision rate (CR) - CA / MS.
- Total moves (TM) - Number of non-still actions taken by all agents.

We train our networks using a single thread and a single GPU for days. We train curriculum 1 and curriculum 2 for 5000 episodes each, and train curriculum 3 until convergence. For the discount, we set that $\gamma = 0.95$. For the ϵ-greedy policy, we set that $\epsilon = 0.05$. Moreover, we use a learning rate of 5e-4, a batch size of 32, and a replay buffer size of 5000.

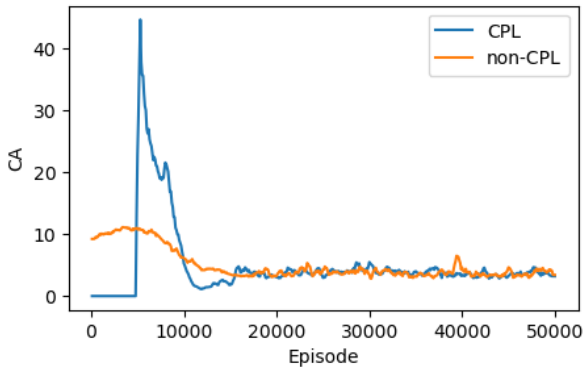
B. Results Analysis

Table.II shows the comparison results of CPL and baselines on 6 measurements under different settings of obstacle density and number of agents. The number of agents varies in four cases: 8, 16, 32, and 64, and the density of obstacles varies in four cases: 0%, 10%, 20%, and 30%. The measurements with $\uparrow$ stand that bigger is better, and the measurements with $\downarrow$ stand that smaller is better. In each column, the bolded numbers stand for the best results over all methods.

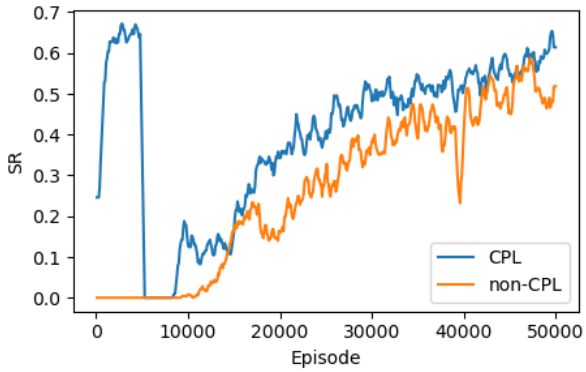
In terms of CA, the number of agent collisions of CPL was close to PICO most of the time and significantly lower than PRIMAL, suggesting that the collision avoidance skills trained in curriculum 2 still work in the later stage of training. In terms of CO, the number of obstacle collisions of CPL is consistently 0, as the action space in CPL states that collisions with static obstacles are illegal actions. In terms of SR, CPL has a significantly higher success rate than the other two baselines, especially in environments with large number of agents and high obstacle density. The key to success in complex environments is the simple-to-hard learning approach of CPL, which is in line with the human intuition for learning to complete a difficult task. In terms of MS, the solutions of CPL always have higher quality, i.e. agents are able to complete path-finding tasks more quickly. This is because the goal of maximising cumulative team rewards as defined in CPL is more in line with the optimisation goal of MAPF tasks than other baselines based only on individual rewards. In terms of CR, CPL performs mediocrely, because the ratio relationship between the number of collisions and the time makespan is not reflected in the cumulative team rewards. If CR is to be used as an indicator, CPL will need to incorporate this consideration in the future. In terms of TM, which measures the total overhead of agent movement in practice, CPL also performs well.

C. Ablation Studies

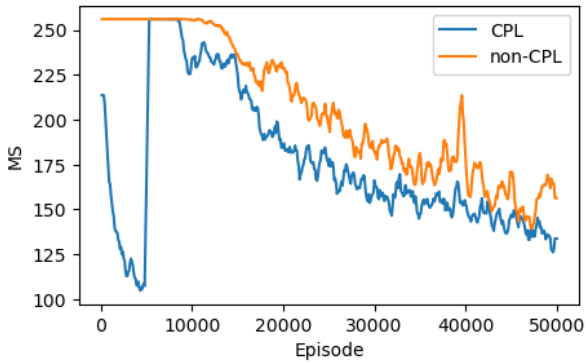
To further analyze the effectiveness of the curriculum learning, we conduct ablation studies to verify the impact of the progressive curriculums on the performance of the algorithm.



(a) Comparison between CPL and non-CPL on CA



(b) Comparison between CPL and non-CPL on SR



(c) Comparison between CPL and non-CPL on MS

Fig. 3. The results of the ablation experiments.

Specifically, we propose a method that does not include the first two curriculums, denoted non-CPL, in which the training process starts directly from curriculum 3 and trains the agent network, the mixing network and the hypernetworks from scratch by team rewards.

We conduct experiments on random grid worlds with a high obstacle density of 30% and a number of agents of 8 to compare the performance of CPL with its ablated version. The comparison results are shown in Fig.3. First, as shown in Fig.3(a), the number of agent collisions of CPL explodes just after the 5000th episode, caused by the transfer from the curriculum 1 single-agent environment to the curriculum

2 multi-agent environment. Thereafter, the numbers of agent collisions of both methods remain at roughly the same level, as they have the same optimization objective in the final stage. Second, as shown in Fig.3(b), the success rate of CPL rises rapidly in the first stage (the 0th-5000th episodes) and also starts to rise around the 3000th episode in the second stage (the 5001th-10000th episodes). Moreover, even after entering the third stage, the success rate of CPL is higher than that of non-CPL with the same total number of episodes, which verifies that the first two curriculums in CPL speed up the learning. Finally, as shown in Fig.3(c), CPL also outperforms non-CPL in makespan, achieving a lower path cost. The results fully verify the significance of the curriculums in CPL for improving training efficiency.

V. Conclusion

This paper proposes CPL based on the CTDE paradigm and the curriculum learning framework to solve MAPF tasks. CPL retains the advantages of learning-based methods, i.e. distributed decision-making, and enables agents to acquire collaborative skills during the training. To counter the exploration problem posed by sparse team rewards and to improve training efficiency, CPL takes advantage of the fact that MAPF has explicit sub-tasks and leverages curriculum learning to transfer from simple sub-tasks to the original task by network parameter inheritance for more complex environments. The training process is divided into three stages, with the training objectives being single-agent path finding, multi-agent path finding and cooperative multi-agent path finding, respectively. Agents learn strategies for completing individual goal sub-tasks from dense individual rewards and then learn strategies for maximising the team return from team rewards. Experiments show that CPL outperforms other state-of-the-art learning-based methods in terms of both success rate and makespan, especially in environments with high obstacle density and agent number. There are also ablation experiments in high obstacle density environments that demonstrate the significance of curriculum learning used in CPL for improving the learning speed.

References

[1] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.

[2] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[3] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[4] R. Morris, C. S. Pasareanu, K. Luckow, W. Malik, H. Ma, T. S. Kumar, and S. Koenig, "Planning, scheduling and monitoring for airport surface operations," in *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of operations research*, vol. 27, no. 4, pp. 819–840, 2002.

[6] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed Autonomous Robotic Systems: The 14th International Symposium*. Springer, 2019, pp. 35–49.

[7] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8699–8705.

[8] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang, "Multi-agent path finding with prioritized communication learning," *arXiv preprint arXiv:2202.03634*, 2022.

[9] B. Freed, G. Sartoretti, and H. Choset, "Simultaneous policy and discrete communication learning for multi-agent cooperation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2498–2505, 2020.

[10] 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.

[11] Y. Zhang, Y. Qian, Y. Yao, H. Hu, and Y. Xu, "Learning to cooperate: Application of deep reinforcement learning for online agv path finding," in *Proceedings of the 19th International Conference on autonomous agents and multiagent systems*, 2020, pp. 2077–2079.

[12] K. Wan, D. Wu, B. Li, X. Gao, Z. Hu, and D. Chen, "Me-maddpg: An efficient learning-based motion planning method for multiple agents in complex environments," *International Journal of Intelligent Systems*, vol. 37, no. 3, pp. 2393–2427, 2022.

[13] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[14] 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," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[15] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[16] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[17] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial intelligence*, vol. 219, pp. 1–24, 2015.

[18] C. Ferner, G. Wagner, and H. Choset, "Odrm* optimal multirobot path planning in low dimensional search spaces," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3854–3859.

[19] 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.

[20] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[21] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 353–12 362.

[22] S.-H. Chan, J. Li, G. Gange, D. Harabor, P. J. Stuckey, and S. Koenig, "Flex distribution for bounded-suboptimal multi-agent path finding," 2022.

[23] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[25] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.