Moving Forward in Formation: A Decentralized Hierarchical Learning Approach to Multi-Agent Moving Together

Shanqi Liu[1], Licheng Wen[1], Jinhao Cui[1], Xuemeng Yang[1], Junjie Cao[1], and Yong Liu[1,*]

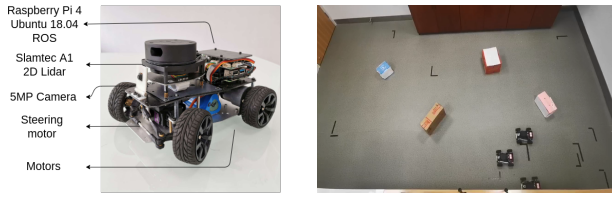
***Abstract*— Multi-agent path finding in formation has many potential real-world applications like mobile warehouse robotics. However, previous multi-agent path finding (MAPF) methods hardly take formation into consideration. Furthermore, they are usually centralized planners and require the whole state of the environment. Other decentralized partially observable approaches to MAPF are reinforcement learning (RL) methods. However, these RL methods encounter difficulties when learning path finding and formation problems at the same time. In this paper, we propose a novel decentralized partially observable RL algorithm that uses a hierarchical structure to decompose the multi-objective task into unrelated ones. It also calculates a theoretical weight that makes each tasks reward has equal influence on the final RL value function. Additionally, we introduce a communication method that helps agents cooperate with each other. Experiments in simulation show that our method outperforms other end-to-end RL methods and our method can naturally scale to large world sizes where centralized planner struggles. We also deploy and validate our method in a real-world scenario.**

I. INTRODUCTION

Mobile robots have been deployed in many real-world applications nowadays, including drone swarm, aircraft-towing vehicles and warehouse robots [1] [2]. In many of these scenarios, it is important for the agents to move in a specific formation while avoiding obstacles [3]. For example, the warehouse robots need to work together to transport large cargo and some search and rescue missions require drone swarms to cross the forest while maintaining formation. However, most current multi-agent path finding (MAPF) algorithms [4], [5] can not plan in such cases as they do not take formation into consideration.

There are few works focused on solving the multi-agent path finding in formation (MAiF) problem [6]. Most of them are centralized algorithms. A centralized planner needs the information and intentions of all agents to generate collision-free paths [7]. It becomes infeasible when the number of agents grows and the map size grows [8]. Besides, we believe that considering a partially-observable world is an essential step towards real-world deployment. Therefore, we focus on decentralized methods that rely on a limited field of view to solve MAiF problem.

Most of the former decentralized partially observable methods to MAPF are reinforcement learning (RL) algorithms [9], [10]. They have shown great potential to learn a general policy using local observation [11]. However, learning different tasks together can be challenging, especially when optimizing conflicting objectives. The majority of multi-objective reinforcement learning (MORL) approaches are single-policy algorithms. They adopt a linear scalarization function in order to learn Pareto optimal solutions [12]. The linear scalarization function is a weighted sum of the parameters that the transform multi-objective rewards vector into a single value. However, the weights used during learning relies on manual design and fine-tuning [13].



(a) WeTech Robot (b) Snapshot of the field test

Fig. 1: We test the proposed method on ackermann-steering robots produced by WeTech. The experiment video is available in the attachment of this paper.

In this paper, we propose a novel hierarchical reinforcement learning algorithm to generalize previous MAPF methods. The major contributions of this paper are summarized as follows:

- We propose a hierarchical reinforcement learning structure to divide the multi-objective task into unrelated ones. We train each task's policy separately by optimizing its own reward.
- We propose a novel way to calculate the linear scalarization function based on the well-trained policies. Our method can calculate the theoretical weights that make each task's reward have equal influence on the final RL value function.
- We introduce a communication method that takes up little bandwidth to help agents cooperate with each other.
- We perform extensive experiments in both simulation and real-world scenarios, where our method outperforms other comparison methods and scales to large world sizes where centralized planners struggle.

II. RELATED WORKS

1) Moving Agents in Formation (MAiF): MAiF, a variant of Multi-Agent Path Finding (MAPF) problem, contains two key sub-tasks: planning collision-free paths for multiple agents while keeping the agents in formation. The former

[1]Shanqi Liu, Licheng Wen, Jinhao Cui, Xuemeng Yang, Junjie Cao and Yong Liu are with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China (*Yong Liu is the corresponding author, yongliu@iipc.zju.edu.cn)

subtask can be addressed by numerous MAPF planner, including reduction-based methods [4], [5], A*-based methods [14]–[16], and dedicated search-based methods [7], [17], [18]. Formation-control algorithms can apply to the later subtasks. A method of planning motion for formations of nonholonomic robots is presented in [19], and a control method for a team of mobile robots maintaining and changing the desired formation using graph theory is presented in [20]. The MAiF problem is formally proposed in [6]. In this work, the authors develop a two-phase search algorithm to solve both sub-tasks simultaneously.

2) Multi-Agent Reinforcement Learning (MARL): The most important problem encountered when training a multi-agent policy is the curse of dimensionality. Most centralized approaches fail as the combination of state-action spaces is an exponential explosion, requiring impractical amounts of training data to converge. Thus, many works have focused on centralized training and decentralized execution(CTDE) policy learning. VDN [21], QMIX [22] and Qtran [23] represent Q learning based CTDE methods. We use VDN to train our policy in this work. There are other methods like MADDPG [24] and COMA [25] bases on actor-critic structure to train a CTDE policy.

3) Hierarchical Reinforcement Learning (HRL): There are several RL approaches to learning hierarchical policies [26]–[28]. However, these have many strict limits and are not off-policy training methods. Recently popular works like HIRO [29], Option-Critic [30] and FeUdal Networks [31] have achieved quite good performance. Especially, HAC [32] uses hindsight [33] to overcome non-stationary that came from continually changing sub-policy in HRL, so it can use off-policy method to have a better performance. However, in our work, we can directly use off-policy method as we already had a well-trained sub-policy.

4) Multi-Objective Reinforcement Learning (MORL): MORL algorithms have two main categories [34], [35]: single-policy methods and multiple-policy methods. Single-policy approaches usually aim to find the optimal policy for a given weight among the objectives [36], [37]. Multi-policy methods aim to learn a set of policies to obtain the approximate Pareto frontier. They usually perform multiple runs of a single-policy method over different preferences [38]–[40].

III. POLICY REPRESENTATION

A. Observation and Basic Action Definition

We consider a partially observable discrete grid graph, where agents can only observe the state of the world in a limited field of view around themselves (9x9 FOV in practice). As shown in Fig. 2, our observation are divided into four channels: *i)* Obstacle map: the obstacle grids equal 1, empty grids equal 0; *ii)* Position map: the grid contains other agents equals the id of that agent, otherwise zero, different color stands for different agents in figure; *iii)* Cost map: the cost of the shortest path from each grid to the agent's goal, this observation is pre-computed before training, different color stands for different cost in figure;

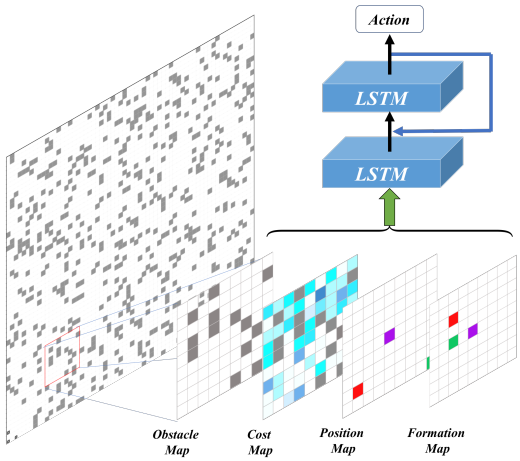


Fig. 2: Observation space of simulation environment and our model structure.

iv) Formation map: it contains the desired formation of all agents.

Agents take five discrete actions in the grid world: moving to one of the four cardinal cells or staying still. At each time step, some actions may be invalid, such as moving into a wall.

B. Formation Loss Function

Inspired by Procrustes Analysis [41], we define our loss function between two formations. We assume k agents in Cartesian plane has two formations $X_1 = ((x_1, y_1), ...(x_k, y_k))$ and $X_2 = ((w_1, z_1), ...(w_k, z_k))$. We define the formation loss L_f between X_1 and X_2 as:

$$L_f(X_1, X_2) = \left\| X_2 - X_1\Gamma - 1_k\gamma^{\mathrm{T}} \right\|^2 \tag{1}$$

where

$$\Gamma = \mathbf{M}(\theta), \theta = \tan^{-1}\left(\frac{\sum_{i=1}^{k}(w_i y_i - z_i x_i)}{\sum_{i=1}^{k}(w_i x_i + z_i y_i)}\right)$$
$$\gamma = \left[\frac{\sum w_i - \sum x_i}{K}, \frac{\sum z_i - \sum y_i}{K}\right]$$

The $\|X\| = \left\{\mathrm{trace}\left(X^{\mathrm{T}}X\right)\right\}^{1/2}$ is the Euclidean norm, and $\mathbf{M}(\theta)$ denotes a 2D rotation matrix with angle θ. The loss function we designed is more reasonable than the function in [6] for they didn't consider the rotation transformation between agents' formations.

C. Policy Definition

We have three policies used in this work: path finding policy, formation policy and meta policy. Each policy has its unique task, which will be introduced below. The overall hierarchical reinforcement learning structure will be discussed in IV-A.

1) Path Finding Policy: Path finding policy aims at solving the MAPF problem without considering the formation loss. In order to reduce the training difficulty, we propose an action clipping method to reduce the action dimensions. The original MAPF problem usually contains swapping conflicts defined as agents planning to swap locations in a single time step. It only occurs when two agents have the opposite goal directions, which is impossible in the MAiF scenario because

all agents in the formation ought to move toward the same goal direction. As all other collisions can be solved by one agent waiting when the other agent moves towards the goal position, we can optimize path finding policy by clipping the non-optimal actions. The non-optimal action is defined as the action, which makes the value increase in the cost map. We define a reward that encourages agents to find the path in Table. I.

2) Formation Policy: The formation policy focus on keeping agents in a specific formation. We design a training environment, which randomly generates agents and keeps all agents within at least one agent's field of view. We also clip the action space by forbidding invalid moves (run into walls). The reward can be seen in Table. I. L_f in the table is the formation loss.

3) Meta Policy: Meta policy is a high-level policy that decides which low-level policy should be used at each step. The reward is stated in Table. I. The w_f is the basic weight of L_f that uses to balance the reward of path finding and keep formation. More details will be discussed in Section. IV-B.

TABLE I: ALL REWARD STRUCTURES

	Path finding	Formation	Meta policy
Agent Collision	-50	-50	-50
Movement Towards Goal	1	0	1
No Movement	-0.25	-0.25	0
Finish Episode	100	0	0
Formation Loss	0	-L_f	-$w_f \cdot L_f$
Keep Formation	0	100	0

All rewards are valued per time steps.

IV. METHOD

In our paper, we propose a novel hierarchical reinforcement learning method that can decompose the path finding and formation problem into unrelated ones. Comparing to end-to-end reinforcement learning methods, our method can significantly reduce the learning difficulty and easily transfer to new situations. Furthermore, we propose a novel approach to calculate the theoretical linear scalarization function, which can balance the path finding and formation policies. Finally, we introduce a communication method that benefits the training process of decentralized cooperative policy.

A. Hierarchical Learning

One of the key challenges in training a well-performed policy of MAiF is that the overall task is complex. The MAPF problem itself is hard to learn since it usually takes hundreds of hours to train [9]. Nevertheless, the formation policy needs to be learned simultaneously. These two goals are conflicting with each other thus the learning process would be unstable. Therefore, learning an end-to-end policy to solve the MAiF problem is difficult. To solve this problem, we propose a novel hierarchical reinforcement learning structure that can decompose the path finding and formation problem into unrelated ones. Then we can train path finding and formation policy separately. By defining unique rewards for each policy, they can learn their own task without being interrupted by optimizing the other task.

However, we have not already solved MAiF problem even if we have a well-trained path finding policy and formation policy. We still need the policy to decide which policy should be used at each time step. Moreover, we can train a meta policy to do so. The meta policy's task is to balance the path finding task and keep formation task by optimizing its own designed reward. As the low policy (path finding and formation policy) is already well-trained, there is no worry about the mutual interference in low policy learning process. However, we still need to design a suitable reward for meta policy to solve multi-objective learning tasks, which will discuss in section IV-B. Nevertheless, as the meta policy has lower dimensions of action and faces a simpler task to learn, optimizing multi-objective tasks can be done much more efficiently. Finally, as all agents are homogeneous, we reuse the policies among all agents. The overall algorithm can be viewed as Algorithm 1.

Algorithm 1 Hierarchical learning

Initialize $\pi_{meta}(a_m|o_t;\theta_m)$,$\pi_p(a_p|o_t;\theta_p)$,$\pi_f(a_f|o_t;\theta_f)$
Initialize three different environments E_m,E_p,E_f
Pretrain π_p and π_f separately in E_p,E_f
Initialize replay buffer R_m
Initialize Q-Network Q_{θ_m} and target Q-Network $Q_{\theta'_m}$

for n episodes = 1 to N **do**
  Agents take meta action $a_m = \pi_m(o)$
  if a_m is using path finding policy **then**
    $a_{low} = \pi_p(o)$
  else
    $a_{low} = \pi_m(o)$
  Store $< o, a_m, r, o' >$ in R_m
  Sample a mini-batch B_m from R_m
  Perform a gradient decent step on
  $(y - Q_{\theta_m}(o,a))^2_{B_m}$,
  where $y = r + \tau Q_{\theta'_m}(o', argmax_{a'_m}(Q_{\theta_m}(o', a'_m)))$.
  if n mod $I_{TargetUpdate}$ **then**
    Update $Q_{\theta'_m} : \theta'_m \leftarrow \theta_m$

B. Multi-Objective Learning

In this work, we use a linear scalarization function to define a utility over a vector-valued reward and thereby reducing the dimensionality of the multi-objective reward vector to a single, scalar value.

The linear scalarization function f is a function that projects a vector v to a scalar: $v_w = f(v, w)$, where w is a weight vector parameterizing f. In our situation, our meta policy needs to balance the formation reward and path finding reward, so our scalarization function f can be viewed as $v_w = f((r_p, r_f), w)$. r_p and r_f stands for the reward of path finding and formation, in this case, w is a two dimensions vector.

Generally, since the boundary policy such as going straight to the goal position without keeping formation can be easily obtained, we hope to find the w to make policy considers maximizing all rewards equally. The values of the w are usually decided manually in previous works, which is infeasible in our situation as we do not know the accurate

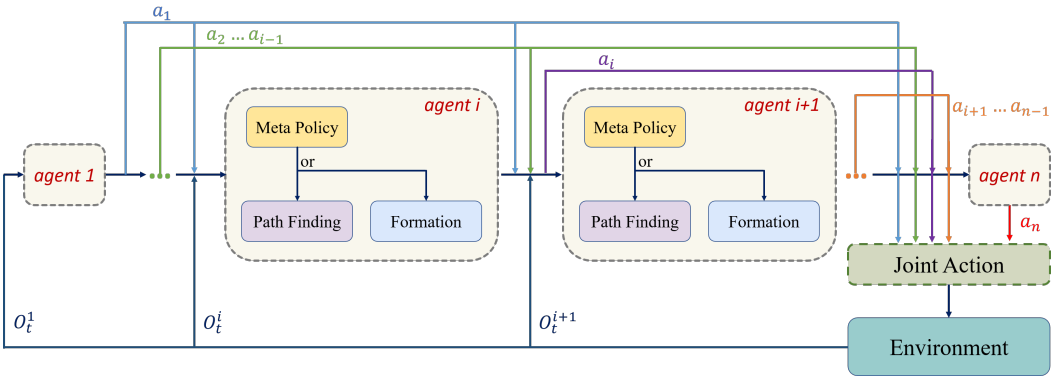


Fig. 3: Structure of our overall policy. Agent 1 is the chosen leader in every time step who acts first. Other agents select actions sequentially based on its local observation and all the former agents' actions. Each agent has the same hierarchical structure that uses a meta policy to choose between the path finding policy and formation policy.

upper limits of our rewards. Benefiting from the structure of hierarchical reinforcement learning, we introduce a novel MORL algorithm. It calculates a theoretical weight that balances path finding policy and formation policy by making all rewards have equal influence on the final RL value function. Then we choose other values around this basic weight to get the Pareto fronts. Notably, the method can not only use in MAiF problem but also solve other MORL problems. Let w_f be the basic w for r_f, L_f be the formation loss and R^* be the range, we have

$$w_f = \frac{T}{R^*(\mathbb{E}[\sum_{t=0}^{T} \Delta L_f(o_t)])} \tag{2}$$

The proof of Eq. 2 can be found in Appendix. VI-A.

C. Decentralized Cooperative Multi-Agent Learning

It is challenging to train a fully decentralized multi-agent cooperative policy, especially when the task needs highly cooperative agents. We use VDN [21] to train agents in a centralized way but can act decentralized. However, cooperative policies can not perform well when it relies on a single agent's observation. To solve this problem, we propose a novel communication method based on theory of mind. The theory of mind (ToM) indicates that agents can interpret others' actions and act in a more informative way. Thus, we use a combination of single agent observation and other agents' joint actions u_t^- as the input for all agents' low-level policies. Our communication method only involves low-dimension action information, which takes up little bandwidth and makes it suitable to deploy in the real world.

However, if every agent's action is based on other agents' actions, we can not calculate all actions at the same time step. In other words, one agent must move first without inferring other agents' actions. Then other agents can select movements based on the performed actions of former ones. In such a situation, the joint action is significantly influenced by the first taken action because all other agents cooperate with the former ones. So how to choose the first move agent, called leader, is essential for improving policy performance.

In our work, we propose a dynamic leader chosen method. When most agents are executing formation policy, we choose the leader as the agent in the middle of the formation. The reason is that the middle agent has the broadest view of all agents' relative positions. It can move to an optimum position in favor of resuming or keeping formation. While most agents are executing the path finding policy, we choose the agent in the front as the leader, for that the leader can observe the future path and choose a path with fewer obstacles to go.

V. EXPERIMENTS

A. Experiment Settings

For our experiments, we chose our three centralized planning methods for comparison: Joint-State A* [18], SWARM-MAPF [6] and Conflict-Based Search (CBS) [17]. The joint-state A* performs poorly on scalability, yet it finds the optimal Pareto fronts. SWARM-MAPF is the state-of-the-art centralized planner which gives a nearly optimal solution to MAiF. We use SW for short in Table. II. CBS can give us a baseline as the pure MAPF method that does not optimize formation loss. For all centralized planning methods except CBS, we used a time limit of 300s. Note that all other centralized methods have access to the whole state of the environment. In contrast, our method assumes that each agent only has partial observability of the environment and plans as an individual. As for other RL methods, to the best of our knowledge, none of the previous work in RL can directly apply to MAiF. So, we compare with VDN [21] and a centralized training centralized executing (CTCE) method as RL baseline methods. The CTCE method takes all agents as one agent, uses the joint observations as input and outputs the joint actions. All RL methods use the same observation form and same network structure. These two compared methods also use action clipping and the basic w_f to balance rewards for fair comparison.

In our experiments, we compare makespan, formation loss, runtime and success rate with centralized planning methods and RL methods. Specifically, makespan means the number of steps to finish an episode by reaching the goal position. The runtime is evaluated from loading maps and creating environment to the end of each episode and formation loss is defined in section III-B. For a fair comparison, our method is tested in untrained maps. The test maps are randomly generated and have various sizes and obstacle densities. For

TABLE II: RESULTS OVER DIFFERENT EXPERIMENT SETTINGS. '-' MEANS THE METHOD FAIL TO GET RESULT.

Environment Setting			Makespan				Formation Loss				Success Rate				Runtime(s)			
map size	agent	d	Ours	CBS	SW	A*	Ours	CBS	SW	A*	Ours	CBS	SW	A*	Ours	CBS	SW	A*
20×20	3	0.15	41.3	**34**	35.4	-	0.5	3.37	**0.12**	-	**1.0**	**1.0**	**1.0**	0.0	0.27	**0.002**	0.05	-
20×20	3	0.05	34.2	**34**	**34**	**34**	0.01	0.2	**0.0**	**0.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.24	**0.0002**	0.002	0.003
20×20	5	0.05	**30**	**30**	**30**	**30**	**0.0**	1.05	**0.0**	**0.0**	0.8	**1.0**	**1.0**	**1.0**	0.39	**0.0004**	0.002	0.003
512×512	3	0.15	1131.2	**1018**	-	-	**0.37**	9.26	-	-	**1.0**	**1.0**	0.0	0.0	**8.47**	171.58	-	-
512×512	4	0.05	1713.6	**1014**	**1014**	-	1.07	10.4	**0.0**	-	**1.0**	**1.0**	**1.0**	0.0	16.3	58.9	**4.5**	-
1024×1024	3	0.05	2378.7	**2042**	-	-	**0.17**	24.7	-	-	**1.0**	**1.0**	0.0	0.0	**18.1**	>300	-	-
1024×1024	3	0.15	**2215.5**	-	-	-	**0.38**	-	-	-	**1.0**	0.0	0.0	0.0	**17.2**	-	-	-

all decentralized partially observable methods, we add a few random steps in the beginning of each testing episode to try different directions of departure so that we can search different paths to find the nearly optimum solution as the final performance. All experiments are carried out on the same computer, equipped with an Intel i7-7700K, 16GB RAM and an NVIDIA GTX1080Ti.

Our benchmark and code will be released in `https://github.com/zijinoier/mater`.

B. Training Details

1) Environment: We apply a grid world simulation environment, just as Fig. 2 shows. The map size is $\{20, 32, 512, 1024\}$. The obstacle density is $\{0.05, 0.15\}$. We make a limit length of walls as half of the length of the agents' view field. This can prevent the agent from being completely separated in the field of view. Otherwise, agents may not be able to find each other. For each map, the top-left 5×5 or 10×10 cells (depends on map size) are possible start locations, and the bottom-right 5×5 or 10×10 cells are possible goal locations. The formation is placed in start locations. During the training process, the environment maps are randomly selected at the beginning of each episode in a map pool with 100 different maps. The map size is 32 and the obstacle density is 0.15. During the testing process, all maps are generated randomly and all algorithms are tested 10 times on different maps.

2) Parameters: : We use a discount factor γ of 0.95, an episode length of 3 times of map size (maximum), and a batch size of 32. We use basic w_f to balance the reward. All policies use the same network consists of two LSTM layers with 256 units. We use the Adam optimizer with a learning rate $2 \cdot 10^{-6}$.

C. Result

Table. II shows the results of our method comparing with centralized methods. All experiments are evaluated in 10 different maps. The formation loss value is normalized by map size. Based on our results, we show that our method is transferable and can be adapted to any map size and obstacle density. Furthermore, we notice that our approach performs extremely well in the large scale world that other centralized methods can not handle. The reason is that our method acts only refer to local observation and the planning time grows linearly as the map size or agent number grows. Our method can handle different obstacle densities without paying extra computing expenses. So, the time complexity of our method is $O(n)$. Meanwhile, the centralized methods' runtime is $O(n^2)$, which grows exponentially as the map size, agent number or obstacle density grows [42]. Therefore, they cannot cope with maps of large size or high obstacle density. In small-sized maps, our method can achieve performance similar to centralized methods, even if our method plans separately based on a limited field of view. However, we notice that our method fails few times when there are five agents in the map, which is because when a certain part of the map gets extremely crowded, it is too hard for agents to avoid collisions.

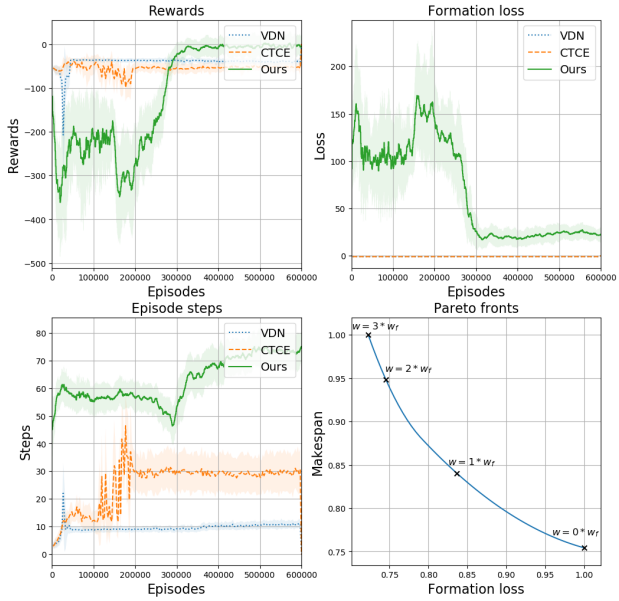


Fig. 4: Left: rewards(top) and episode steps(bottom) of all RL methods. Right: formation loss(top) of all RL method and Pareto fronts(bottom) of our method. For fair comparing, we train both VDN and CTCE three times longer than our method as we have pre-trained path finding and formation policies. In order to facilitate the display, we have scaled our method on the horizontal axis(3x along x). We also normalize the coordinate axis of the Pareto fronts.

We present our method's result compared with end-to-end RL methods and the Pareto fronts of our method in Fig. 4. We notice that other end-to-end RL methods can hardly get out of a ground performance. This can be reflected when evaluating the episode reward defined in Table. I, the CTCE and VDN converge to the collision return, meaning they all fail to reach the goal position. Simultaneously, our method has a lower return at the beginning but learns to avoid collision and reduce the formation loss to converge to the nearly optimal return in the end. We believe that CTCE and VDN are stuck in some kind of local optimal policy because they try to learn both path finding task and formation task together, and optimizing two conflict objectives can cause learning in a dilemma. So they tend to finish the episode by

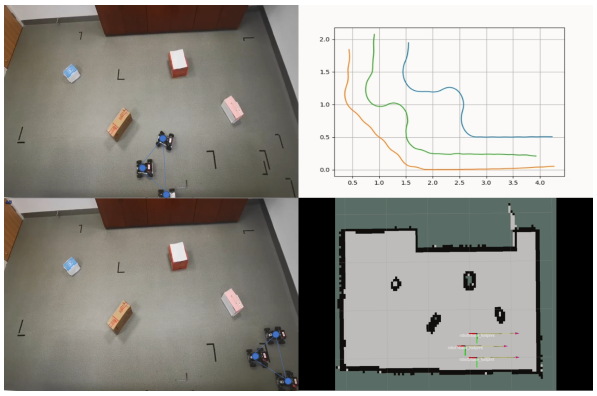


Fig. 5: Snapshot of hardware experiment. Left: moving agents(top) and agents' start positions(bottom), the blue triangle stands for the formation. Right: trajectories(top) and RVIZ view(bottom) of all agents.

an early collision.

As both end-to-end RL methods can not even learn to reach the goal position, we can not compare formation loss with them. However, we notice that our method's formation loss converges to nearly zero after a temporary increase. We believe this is because the meta policy tries to learn to avoid the collision firstly, as the farther the agents are, the less likely they will collide. And when this task is done, it quickly optimizes the formation loss, leading to a rapid increase in episode reward. This result also shows our method can jump out of the local optimum and converge to the global optimum.

When evaluated in episode steps, we notice that the CTCE method can go further than VDN before collision. However, they both fail to reach the goal position without collision happened. Meanwhile, although our method fails to avoid collision in early episodes, our method manages to reach the goal position without collision at the end of the training process. This result also shows the CTCE policy has a better performance than VDN. This is due to the fact that its agents can obtain information from other agents while selecting actions. This information can improve the performance of cooperative policy in the same way as our communication method.

Finally, we present our Pareto fronts. We notice that our method reaches a policy that can balance path finding and keeping in formation when the weight of formation loss equals our base weight. This is in line with our theoretical calculations. We can get the whole Pareto fronts by using n times base weight (e.g. 0,2,3 in the figure).

D. Hardware Experiments

We also implement our method on a small fleet of Ackermann cars. Due to venue restrictions, we choose an indoor room to simulate a three cars formation. Each car plans on the host computer using our decentralized approach. And we transform the discrete action into continual action so the Ackermann cars can carry out. The result indicates that our method has clear sim-to-real capabilities, as the planning time per step is below 0.1s on a laptop computer. Fig. 5 shows our Ackermann cars and experiment environment.

VI. CONCLUSIONS

In this paper, we present a new decentralized partially observable approach to multi-agent in formation. It utilizes a novel hierarchical reinforcement learning structure that can solve multi-objective reinforcement learning problems effectively. Furthermore, we propose a theoretical weight calculation method that makes every task's reward has equal influence on the final RL value function. Additionally, we introduce a communication method that helps agents cooperate with each other. Through an extensive set of experiments, we show that our method outperforms several end-to-end RL algorithms and can scale to various formations, world sizes and obstacle densities. Our method performs well in large-scale worlds where centralized methods struggle. Finally, we present a demonstration where we deploy our method in real-world robots, showing our method sim-to-real ability. However, our method still needs to retrain the formation policy when the formation changes. Our future work will focus on finding a general formation policy that can deal with multiple formations.

APPENDIX

A. PROOF

$Proof$: Firstly, we can have the optimal value at a state is given by the state-value function

$$V^*(o_t) = \max_{\pi} \mathbb{E}\left[\sum_{t=0} \gamma^t f(R(o_t))\right] \tag{3}$$

where

$$R(o_t) = \max_{\pi} R(o_t, a_t)$$

Given a particular set of weights $\mathbf{w}$, we substitute scalarization function f into Eq. 3 to obtain

$$V^*(o_t \mid \mathbf{w}) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \sum_{a_t \in A} \gamma^t (w_1 r_1(o_t, a_t) + \cdots + w_n r_n(o_t, a_t))\right] \tag{4}$$

We substitute w_p, w_f, r_p and r_f into Eq. 4 to obtain

$$V^*(o_t \mid \mathbf{w}) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \sum_{a_t \in A} \gamma^t (w_p r_p(o_t, a_t) + w_f r_f(o_t, a_t))\right] \tag{5}$$

Here, if we want to normalize the r_p and r_f, we can let $V^*(o_t \mid \mathbf{w})$ stay the same value if we take optimum a_t whether to max r_p or r_f, we note the optimum a_t^* as a_p^* and a_f^* separately.

$$\begin{aligned} &\Delta V^*(o_t \mid \mathbf{w}) = \\ &\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(w_p r_p\left(o_t, a_p^*\right) + w_f r_f\left(o_t, a_p^*\right)\right)\right] - \\ &\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(w_p r_p\left(o_t, a_f^*\right) + w_f r_f\left(o_t, a_f^*\right)\right)\right] \end{aligned} \tag{6}$$

Since we have already known $r_p(o_t, a_t)$ is 1 only when a_t is a_p. Eq. 6 can be simplified as

$$\Delta V^*\left(o_t \mid \mathbf{w}\right) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(1 + w_f r_f\left(o_t, a_p^*\right) - w_f r_f\left(o_t, a_f^*\right)\right)\right] \tag{7}$$

We can find here that if we want our $\Delta V^*\left(o_t \mid \mathbf{w}\right)$ equals 0, we can just make

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(1 + w_f r_f\left(o_t, a_p^*\right) - w_f r_f\left(o_t, a_f^*\right)\right)\right] = 0 \tag{8}$$

Eq. 8 is only related with reward function r_f. And we know r_f stands for formation loss L_f, we have

$$r_f\left(o_t, a_t\right) = L_f(o_{t+1}) \tag{9}$$

we substitute Eq. 9 into Eq. 8 and take γ equals 1 for simplify

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \left(1 + w_f\left(L_f(o_{t+1}^-) - L_f(o_{t+1}^*)\right)\right)\right] = 0 \tag{10}$$

in which $L_f(o_{t+1}^-)$ means the worst formation loss(comparing to last time step) for time step o_{t+1} and $L_f(o_{t+1}^*)$ is the best one. If we define

$$\Delta L_f(o_t) = L_f(o_{t+1}) - L_f(o_t) \tag{11}$$

We have

$$\begin{aligned} \max(\Delta L_f(o_t)) &= L_f(o_{t+1}^*) - L_f(o_t) \\ \min(\Delta L_f(o_t)) &= L_f(o_{t+1}^-) - L_f(o_t) \end{aligned} \tag{12}$$

So,

$$\begin{aligned} \max(\Delta L_f(o_t)) - \min(\Delta L_f(o_t)) = \\ L_f(o_{t+1}^*) - L_f(o_{t+1}^-) \end{aligned} \tag{13}$$

Then, we substitute Eq. 13 into Eq. 10 :

$$\begin{aligned} &\mathbb{E}\left[\sum_{t=0}^{\infty} \left(1 + w_f\left(L_f(o_{t+1}^-) - L_f(o_{t+1}^*)\right)\right)\right] \\ &= T + w_f(\min \mathbb{E}[\sum_{t=0}^{T} \Delta L_f(o_t)] - \max \mathbb{E}[\sum_{t=0}^{T} \Delta L_f(o_t)]) = 0 \end{aligned} \tag{14}$$

Here, we transfer the problem to estimate the expectation of $\sum_{t=0}^{T} \Delta L_f(o_t)$. This can be easily estimated by evaluating a well-trained formation policy and a random policy, let R^* stands for range here.

$$w_f = \frac{T}{R^*(\mathbb{E}[\sum_{t=0}^{T} \Delta L_f(o_t)])} \tag{15}$$

B. FORMATION SETTING

The formation we used in experiments are shown in Table. III, where 1 in the tables means the agents and 0 means empty space. From left to right are 3 agents' formation(left) and 4 agents' formation(middle) and 5 agents' formation(right).

TABLE III: FORMATIONS

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	1	0	1	0
0	0	0	0	0

0	0	0	0	0
0	1	0	1	0
0	0	0	0	0
0	1	0	1	0
0	0	0	0	0

0	0	1	0	0
0	0	0	0	0
1	0	1	0	1
0	0	0	0	0
0	0	1	0	0

REFERENCES

[1] 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.

[2] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[3] H. Ma, J. Yang, L. Cohen, T. Kumar, and S. Koenig, "Feasibility study: Moving non-homogeneous teams in congested video game environments," *arXiv preprint arXiv:1710.01447*, 2017.

[4] J. Yu and S. LaValle, "Planning optimal paths for multiple robots on graphs," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3612–3617.

[5] P. Surynek, "Reduced time-expansion graphs and goal decomposition for solving cooperative path finding sub-optimally," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[6] J. Li, K. Sun, H. Ma, A. Felner, T. S. Kumar, and S. Koenig, "Moving agents in formation in congested environments," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 726–734.

[7] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Eighth annual symposium on combinatorial search*. Citeseer, 2015.

[8] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 477–483.

[9] 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.

[10] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *arXiv preprint arXiv:2005.05420*, 2020.

[11] 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," *arXiv preprint arXiv:2007.15724*, 2020.

[12] H. Ching-Lai and S. M. M. Abu, *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag, 1979.

[13] A. Kusari and J. P. How, "Predicting optimal value functions by interpolating reward functions in scalarized multi-objective reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7484–7490.

[14] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 3260–3267.

[15] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems." in *AAAI*, vol. 1. Atlanta, GA, 2010, pp. 28–29.

[16] 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.

[17] 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.

[18] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Tenth Annual Symposium on Combinatorial Search*, 2017.

[19] T. D. Barfoot and C. M. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 65–78, 2004.

[20] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[21] 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.

[22] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.

[23] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:1905.05408*, 2019.

[24] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[25] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926*, 2017.

[26] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[27] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in neural information processing systems*, 2016, pp. 3675–3683.

[28] B. Bakker and J. Schmidhuber, "Hierarchical reinforcement learning with subpolicies specializing for learned subgoals." in *Neural Networks and Computational Intelligence*. Citeseer, 2004, pp. 125–130.

[29] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 3303–3313.

[30] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[31] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1703.01161*, 2017.

[32] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *arXiv preprint arXiv:1712.00948*, 2017.

[33] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[34] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 636–14 647.

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

[36] G. Tesauro, R. Das, H. Chan, J. Kephart, D. Levine, F. Rawson, and C. Lefurgy, "Managing power consumption and performance of computing systems using reinforcement learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 1497–1504.

[37] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning," in *Advances in Neural Information Processing Systems*, 2002, pp. 1563–1570.

[38] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[39] M. Pirotta, S. Parisi, and M. Restelli, "Multi-objective reinforcement learning with continuous pareto frontier approximation," in *29th AAAI Conference on Artificial Intelligence, AAAI 2015 and the 27th Innovative Applications of Artificial Intelligence Conference, IAAI 2015*. AAAI Press, 2015, pp. 2928–2934.

[40] S. Parisi, M. Pirotta, and J. Peters, "Manifold-based multi-objective policy search with sample reuse," *Neurocomputing*, vol. 263, pp. 3–14, 2017.

[41] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, no. 1, pp. 33–51, 1975.

[42] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.