2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9340876

MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments

Zuxin Liu[1], Baiming Chen[2], Hongyi Zhou[1], Guru Koushik[1] , Martial Hebert[3] and Ding Zhao[1,*]

***Abstract*— Multi-agent navigation in dynamic environments is of great industrial value when deploying a large scale fleet of robot to real-world applications. This paper proposes a decentralized partially observable multi-agent path planning with evolutionary reinforcement learning (MAPPER) method to learn an effective local planning policy in mixed dynamic environments. Reinforcement learning-based methods usually suffer performance degradation on long-horizon tasks with goal-conditioned sparse rewards, so we decompose the long-range navigation task into many easier sub-tasks under the guidance of a global planner, which increases agents' performance in large environments. Moreover, most existing multi-agent planning approaches assume either perfect information of the surrounding environment or homogeneity of nearby dynamic agents, which may not hold in practice. Our approach models dynamic obstacles' behavior with an image-based representation and trains a policy in mixed dynamic environments without homogeneity assumption. To ensure multi-agent training stability and performance, we propose an evolutionary training approach that can be easily scaled to large and complex environments. Experiments show that MAPPER is able to achieve higher success rates and more stable performance when exposed to a large number of non-cooperative dynamic obstacles compared with traditional reaction-based planner LRA* and the state-of-the-art learning-based method.**

I. INTRODUCTION

Driven by the need for flexible and efficient manufacturing, an increasing number of affordable mobile robots are expected to be deployed in warehouse environments for transportation purposes. One key component to support the applications of large scale robots is the multi-agent path planning technology. Many research efforts have been devoted to this field in recent years from different perspectives.

Generally, multi-agent planning methods can be classified into two categories: centralized methods and decentralized methods. When all the moving agents' intentions (e.g. future trajectories, goals) are known in a static environment, a centralized planner could generate collision-free paths for all the agents [1]. However, the computational burden may be a significant concern as the number of agents grows, and the agent's performance may degrade when exposed to unknown dynamic objects [2]. Besides, in practice, centralized methods heavily rely on stable and fast communication networks and powerful servers, which would be costly to be deployed in large scale environments with a large number of robots. Therefore, in this paper, we focus on decentralized methods, where reliable communication can not be established between agents.

For decentralized methods, each agent independently makes decisions based on its own local observations and policies. A natural question is: what should the agent know and assume about other agents or dynamic obstacles around it? Some approaches assume all obstacles are static and replan at a high frequency to avoid collision [3], while other people assume homogeneous policy for agents and constant velocity for dynamic obstacles [4]. However, we argue that in practice, it is difficult to perfectly estimate neighbouring decision-making agents' intentions without communication. Therefore, instead of using traditional path planning procedures, some recent approaches use reinforcement learning to solve robot the navigation problem by implicitly learning to deal with such interaction ambiguity with surrounding moving obstacles [5], [6], [7], [8].

Though learning-based approaches have shown great potential to model complex interactions in dynamic environments, most of them make assumptions about the homogeneity or motion models of surrounding moving obstacles [9], [10]. In this paper, we focus on planning in mixed dynamic environments where moving obstacles can either be cooperative or non-cooperative. Also, inspired by state-of-the-art trajectory prediction methods [11], we propose an image-based spatial-temporal dynamic obstacle representation, which doesn't need explicit motion estimation and can be generalized to the arbitrary number of agents.

Reinforcement learning agent is usually difficult to achieve satisfying performance in long-horizon tasks with sparse rewards, as in the long-range goal-conditioned navigation problem [12]. Therefore, one insight in this paper is using mature planning methods to guide the reinforcement learning-based local planning policy. In this way, agents can learn from complicated local interaction with dynamic obstacles while persistently moving towards a long-range goal. In addition, to ensure the multi-agent training stability and performance, we propose an evolutionary reinforcement learning method that can be easily scaled to large and complex environments.

The major contributions of this paper are:

1) We investigate the multi-agent path planning problem in mixed dynamic environments without the homogeneity assumption. To model the dynamic obstacles' behavior, we propose an image-based representation which improves our agents' robustness to handle different types of dynamic obstacles.

* Corresponding author: Ding Zhao. Email: (dingzhao@cmu.edu)
[1]Department of Mechanical Engineering, Carnegie Mellon University, USA.
[2]School of Vehicle and Mobility, Tsinghua University, China.
[3]The Robotics Institute, Carnegie Mellon University, USA.

2) We decompose a difficult long-range planning problem into multiple easier waypoint-conditioned planning tasks with the help of mature global planners. Experiments show that this approach can greatly improve the performance of reinforcement learning-based methods.
3) We propose an evolutionary multi-agent reinforcement learning approach that gradually eliminate low-performance policies during training to increase training efficiency and performance.

The structure of this paper is as follows. Section II introduces related works about multi-agent path planning in dynamic environments. Section III provides some preliminaries for our problem formulation. The detail of our multi-agent path planning with the evolutionary reinforcement learning (MAPPER) approach is presented in section IV, followed by section V which shows the experiment results of MAPPER in various grid world simulations with mixed dynamic obstacles. Finally, we give a brief conclusion in section VI.

II. Related Work

A. Decentralized Path Planning in Dynamic Environment

Decentralized planning methods can be broadly classified into two categories: reaction-based and trajectory-based. For reaction-based approaches, we need to specify avoidance rules for dynamic obstacles and re-plan at each step based on the rules, such as D* lite [3] and velocity obstacle (VO) based methods [13], [14]. Trajectory-based approaches usually estimate surrounding dynamic objects' intentions and then search collision-free paths in the space-temporal domain [15]. These methods either require perfect information of surrounding dynamic obstacles (e.g. velocities and positions) or assume that all the moving agents adopt the same planning and control policy so that they are homogeneous [16], [4]. However, such assumptions may not hold in many real-world applications when involved with sensing uncertainty and heterogeneous moving obstacles, such as pedestrians. Beside, increasing local interaction complexity may lead to oscillation behaviors or *freezing robot problem* [17]. Also, in practice, VO-based and trajectory-based approaches usually have several components to process sensor data, such as object-intention estimation and trajectory prediction. Each component may have many hyper-parameters that are sensitive to environment settings, which need extra human efforts to tune. In order to reduce the amount of hand-tuning parameters and deal with sensing uncertainties, some researchers proposed learning-based methods to solve the planning problem.

B. Reinforcement Learning-based Planning

Reinforcement learning-based collision-avoidance algorithms for the single-agent case have been extensively studied in recent years. Deep neural networks are usually used to approximate agent's policy and value function. Some people propose to learn the navigation policy in a completely end-to-end fashion, which directly maps raw sensor data to the agent's action [18], [19]. However, we believe that extracting object-level representation can improve the policy generalization ability, because different sensor data sources may encode the same object-level information. Chen et al. first estimate dynamic obstacle's physical states (e.g. velocity and positions) under certain motion model assumptions, and then feed them into neural network to obtain future actions [5], [10] . However, the agents' number is restricted and can hardly be deployed in large scale environments. [6] addresses the problem of a varying number of dynamic obstacles with LSTM and removes the homogeneity assumption for surrounding agents. However, it still requires explicit estimation of surrounding agents' states and suffers performance degradation in tasks with a large number of agents. For multi-agent case, PRIMAL [8] is the most similar work with ours, which also uses image-based representation and target goal as input sources. However, non-cooperative dynamic obstacles and temporal information are not considered in their work. Besides, their centralized training approach takes a long time even with the help of imitation learning. Our approach differs from theirs in that: 1) we encode both spatial and temporal information of surrounding obstacles in observation representations; 2) we consider planning in mixed dynamic environments; 3) we propose a decentralized evolutionary training method which can converge much faster and can be generalized to arbitrary number of training agents; 4) we use mature global planner to guide the local policy to solve long-range navigation problem. We will use the reinforcement learning method in PRIMAL as an experiment baseline in section V.

III. Background

A. Problem Formulation

We model the multi-agent planning problem under the Markov decision processes (MDPs) framework. An N-agent MDPs can be represented by the state space $\mathcal{S}$, which describes all the possible state configurations of the agents, the action space $\mathcal{A}_1, ..., \mathcal{A}_N$, which defines each agent's possible actions, and the observation space for each agent $\mathcal{O}_1, ..., \mathcal{O}_N$. In this paper, we consider the partially observable situation, which means each agent can not observe all the state. The agent i receives its own observation $\boldsymbol{o}_i : \mathcal{S} \mapsto \mathcal{O}_i$ and produces an action from its stochastic policy $\boldsymbol{\pi}_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, where the policy is parameterized by θ_i. All the agents' actions will produce new state that follows the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_N \mapsto \mathcal{S}$. For each time step, agent i will receive rewards based on state and its action $r_i : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The initial states is determined by the distribution $\rho : \mathcal{S} \mapsto [0, 1]$. The objective is to maximize the expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ of each agent i, where γ represents the discount factor and T is the time length. The detail representations of observation space, action space and rewards will be introduced in section IV.

B. Advantage Actor Critic Algorithm

We use advantage actor-critic (A2C) method [20] as the basis of our multi-agent evolutionary reinforcement learning framework to solve the planning problem. A2C uses

stochastic policy, which is essential in our multi-agent scenario because the equilibrium policies in multi-agent MDPs are usually stochastic [21]. Additionally, policy gradient-based methods usually have better convergence property than value-based methods [20]. The objective of A2C is to find the policy $\boldsymbol{\pi}_\theta(a|\boldsymbol{o})$ that can maximize the expected return $\mathbb{E}_{\pi_\theta} R(\tau) = \mathbb{E}_{\pi_\theta} \sum_{t=0}^{T} \gamma^t r(\boldsymbol{o}_t, a_t)$ over the episode $\tau = (\boldsymbol{o}_0, a_0, \ldots, \boldsymbol{o}_T, a_T)$, where a is the action and $\boldsymbol{o}$ is the observation. Given this objective function, the policy gradient can be computed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|\boldsymbol{o}) R(\tau)] \tag{1}$$

To reduce the gradient variance, we employ a value function $V^{\boldsymbol{\pi}_\theta}(\boldsymbol{o})$ as the baseline and replace the expected return $R(\tau)$ with an advantage function $A^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}, a)$. Then we can rewrite the gradients as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|\boldsymbol{o}) A^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}, a)], \tag{2}$$

where the advantage function $A^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}, a)$ has an unbiased estimation:

$$A^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}, a) = \mathbb{E}_{\boldsymbol{\pi}_\theta}(r + \gamma V^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}')|\boldsymbol{o}, a) - V^{\boldsymbol{\pi}_\theta}(\boldsymbol{o}) \tag{3}$$

The policy $\boldsymbol{\pi}_\theta$ is usually termed as the *actor* to produce actions based on current observations, and the value function $v^{\boldsymbol{\pi}_\theta}$ is the *critic*, which is used to estimate the advantage function $A^{\boldsymbol{\pi}_\theta}$ that indicates the quality of produced actions. In this paper, we approximate the policy and value function with neural networks, which will be introduced in section IV-D.

IV. Approach

This section shows how the multi-agent path planning problem is modeled into an evolutionary reinforcement learning framework. We firstly introduce the observation representation, action space, and reward design of each agent. Then, we detail the model architecture and training procedures.

A. Observation Representation

In many real-world mobile robot applications, people usually use the single beam LiDAR for localization and obstacle detection purposes, which is cheap and reliable [22], [23]. A common map representation based on the LiDAR data is called the cost map, which discretizes a 2D map into a fixed resolution grids and assigns a cost to each grid [24]. The cost and obstacle information can be continuously updated by the local observation of sensor data. Therefore, to mimic such common map representations in practice, we consider a partially observable grid world environment, where each agent has its own visibility that limited by the sensing range and there is no communication between agents. We argue that such a fully decentralized partially observable setting is feasible and important if we need to deploy our approach to the real-world with large scale robots. We assume that each agent is able to detect and distinguish surrounding agents and dynamic objects within its sensing range and estimate their relative positions. Also, we assume that each agent can access the static environment map so that it can plan a trajectory in this map.

We split the observations into three channels to encode different types of information. As shown in Fig. 1, the first channel stores current observed static obstacles, surrounding agents and dynamic objects' positions, which are represented by different values. This channel is the basic reflection of sensing data and is corresponding to the cost map representation, which could be used in many traditional search-based planning algorithms [3]. The second channel is the trajectory of surrounding agents and dynamic obstacles, which encodes the time sequence information. Inspired by the state-of-the-art trajectory prediction method in the autonomous vehicle field, we encode the trajectory with different grayscales in time [11]. For example, the point on a trajectory in the earlier time has a smaller value than the later one. The third channel is the reference path planned by a global planner based on the static environment map. The reference path update frequency could be much lower than our reinforcement learning-based local planner. We will demonstrate the importance of those observation representations in the experiment section V.

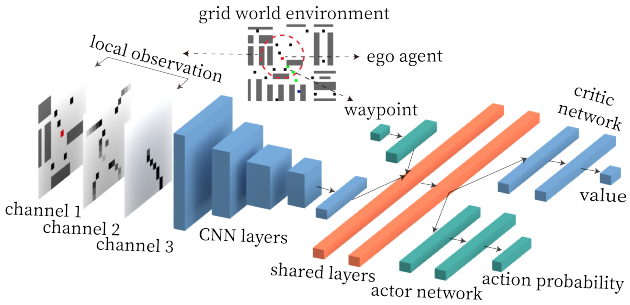

Fig. 1: MAPPER model architecture overview.

B. Action Space

In this paper, we consider an 8-connected grid environment, which means the agent can move to 8 directions: south, north, west, east, southwest, northwest, southeast and northeast. The agent can also choose to wait at the current grid. Thus, the action space contains 9 discrete options in total. At each time step, the agent will move one step following the direction that is selected. However, if the target grid is already occupied, the agent will not be able to move and will stay at the current position.

C. Reward Design

The objective of robot navigation is to reach the goal position with minimum steps while avoiding collision with obstacles. Therefore, the first part of the reward consists of step penalty r_s, collision penalty r_c and goal-reaching reward r_g. To encourage exploration, we penalize slightly more for waiting than moving if the agent has not reached the goal. A similar training trick is also used in [8]. To prevent agents from adopting oscillating policies, we set penalty r_o to agents that return to the positions they come from last

time. The detailed values of these reward components in our experiment can be found in Table I.

Since our local planning policy is guided by a reference path planned by global planner, we introduce an additional off-route penalty r_f if the agent deviates from the reference path. The intuition is that if there are no dynamic obstacles around the agent, it should be able to follow the reference path. To obtain the off-route penalty, we need to calculate the Euclidean distance between the agent's position and the closest point's position on the reference path. Denote the position of the agent as $\boldsymbol{p}_a \in \mathbb{R}^2$. Denote the reference path as a set of coordinates $\mathcal{S} = \{\boldsymbol{p}_{start}, ..., \boldsymbol{p}_{goal}\}$, the penalty is calculated by $r_f = -\min_{\boldsymbol{p}\in\mathcal{S}} ||\boldsymbol{p}_a - \boldsymbol{p}||_2$. Then the final reward is $R = r_s + r_c + r_o + r_g + \lambda r_f$, where the λ controls the weight of off route reward.

TABLE I: Reward Design

Reward	Value
step penalty r_s	-0.1 (move) or -0.5 (wait)
collision penalty r_c	-5
oscillation penalty r_o	-0.3
off-route penalty r_f	-$\min_{\boldsymbol{p}\in\mathcal{S}} ||\boldsymbol{p}_a - \boldsymbol{p}||_2$
goal-reaching reward r_g	30

Algorithm 1 Multi-Agent Evolutionary Training Approach

Require: Agents number N; discount factor γ; evolution interval K; evolution rate η;

Initialize agents' model weights $\Theta = \{\Theta_1, ..., \Theta_N\}$
repeat
  Set accumulated reward $R_1^{(k)}, ..., R_N^{(k)} = 0$
  // update model parameters via A2C algorithm
  for $k = 1, ..., K$ **do**
    for each agent i **do**
      Executing the current policy $\boldsymbol{\pi}_{\Theta_i}$ for T timesteps, collecting action, observation and reward $\{a_i^t, \boldsymbol{o}_i^t, r_i^t\}$, where $t \in [0, T]$
      Compute return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$
      Estimate advantage $\hat{A}_i = R - V^{\boldsymbol{\pi}_{\Theta_i}}(\boldsymbol{o}_i)$
      Compute gradients $\nabla_{\Theta_i} J = \mathbb{E}[\nabla_{\Theta_i} \log \boldsymbol{\pi}_{\Theta_i} \hat{A}_i]$
      Update Θ_i based on gradients $\nabla_{\Theta_i} J$
    end for
    $R_i^{(k)} = R_i^{(k)} + R_i$
  end for
  Normalize accumulated reward to get $\bar{R}_1^{(k)}, ..., \bar{R}_N^{(k)}$
  Find maximum reward $\bar{R}_j^{(k)}$ with agent index j
  // Evolutionary selection
  for each agent i **do**
    Sample m from uniform distribution between $[0, 1]$
    Compute evolution probability $p_i = 1 - \frac{\exp(\eta \bar{R}_i^{(k)})}{\exp(\eta \bar{R}_j^{(k)})}$
    if $m < p_i$ **then**
      $\Theta_i \leftarrow \Theta_j$
    end if
  end for
until converged

D. Model Architecture

We use deep neural networks to approximate the policy and the value function in our A2C method. The model architecture is illustrated in Fig. 1. We have two input sources to be processed independently before being concatenated as a combined feature. The first one is the three channels image represented observation, which has been introduced in section IV-A. The image channels are passed through several blocks, which contain convolution layers, and max-pooling layers. After the final block, the extracted feature will be flattened to one feature embedding.

We notice that reinforcement learning may hardly solve long-term tasks to get the reaching goals rewards [12]. Therefore, instead of using final goals as one input source, we use the waypoint coordinates as sub-goals of our task, which is computed by the global planner. More specifically, the global planner, which is the A* planner in our case, will generate a reference path from the start point to the goal. Then the agent will choose waypoints on the reference path based on a certain distance interval threshold and attempt to reach them one by one. Once the agent approaches its current waypoint goal within a pre-defined range, it will begin to head to the next waypoint.

The currently selected waypoint can be viewed as a sub-goal. It will be passed through a fully connected layer, and then be fed together with the observation feature embedding to two shared fully connected layers. The output feature of the two shared layers will then be passed through two separate neural networks. The lower one is a two-layers policy network with softmax activation, which produces the probability of choosing each action. The upper one is the value function network, which outputs the expected value of the current state.

E. Multi-Agent Evolutionary Reinforcement Learning

Although reinforcement learning has achieved great success in many single-agent tasks [25], it is still hard to directly apply those methods to the multi-agent case. One challenge is the scalability issue: as the number of agents grows, the environment becomes more complicated and the variance of policy gradients may grow exponentially [26].

Inspired by evolutionary algorithm that has been successfully applied to many optimization problems [27], we adopt a decentralized evolutionary approach based on A2C algorithm, which can be applied to arbitrary number of agents training procedure. Evolutionary algorithm usually contains three stages: crossover, mutation and selection. Let's denote the model parameters of agent i as Θ_i. We firstly initialize N agents with random weights for their own model. Then the mutation process begins by training each agent's model separately using A2C algorithm. After k episodes training, agent i will accumulate the rewards over the last k episodes, and we denote it as $R_i^{(k)}$. Denote $R_{max}^{(k)} = \max_{i\in\{1,...,N\}} R_i^{(k)}$ and $R_{min}^{(k)} = \min_{i\in\{1,...,N\}} R_i^{(k)}$. We normalize the accumulated reward for agent i by: $\bar{R}_i^{(k)} = \frac{R_i^{(k)}}{R_{max}^{(k)} - R_{min}^{(k)}}$. Assume agent j has the maximum normal-

ized reward $\bar{R}_j^{(k)} = \max_{i\in\{1,\ldots,N\}} \bar{R}_i^{(k)}$, then we start the crossover and selection stages. Each agent i has the probability p_i to reserve its original model weights and $1-p_i$ probability to replace its weights with the weights of agent j. The probability is calculated by $p_i = 1 - \frac{\exp(\eta \bar{R}_i^{(k)})}{\exp(\eta \bar{R}_j^{(k)})}$. where η controls the evolution rate. Larger η means agents with lower rewards are more likely to be updated. The core idea of our evolutionary method is very simple: gradually eliminate bad policies while maintaining good ones. The full MAPPER training process is shown in Algorithm 1.

V. Experiment and Discussion

A. Experiment Settings

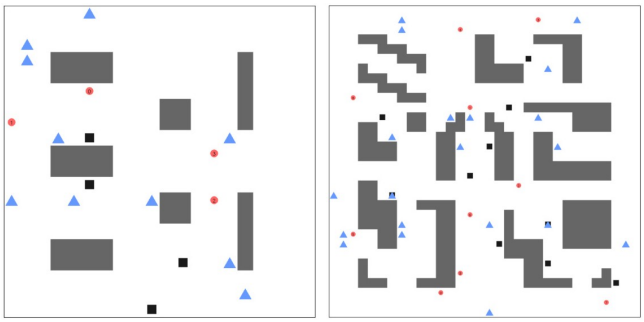

Fig. 2: Grid world simulation environment demonstration.

We evaluate our approach in grid world simulation environment, just as Fig. 2 shows. Gray blocks are static obstacles and black blocks are agents' goals. Orange circles represent agents. Each agent has a 7-grid sensing range in our experiment setting, which means the size of the observation image is $15 \times 15 \times 3$. Blue triangles present dynamic obstacles, where each dynamic obstacle will navigate to a randomly selected goal using LRA* algorithm [28]. To increase the dynamic obstacle movement pattern diversity, we randomly select 50% dynamic obstacles that will ignore the presence of surrounding agents, which would be more challenging for our agents because of their non-cooperative nature.

Existing centralized multi-agent path planning methods, such as conflict based search [1], break down in mixed dynamic environments because of the unpredictable nature of non-cooperative moving obstacles. Therefore, we resort to decoupled reaction-based planning approaches. One benchmark we adopt is a modified local repair A* (LRA*) algorithm that re-plans at every time step, where we replace A* with D* lite [3] implementation because the latter is more computationally efficient in dynamic environments [28]. Each LRA* agent takes into account local observation, updates the cost map accordingly, and searches for a route to the destination. Then, a coordinator that has access to every agent's future plan resolves conflicts between agents and adjust these paths. LRA* behaves similarly to our MAPPER method in that they both react based on local observation, but note that we do not require access to all agents' future plan information.

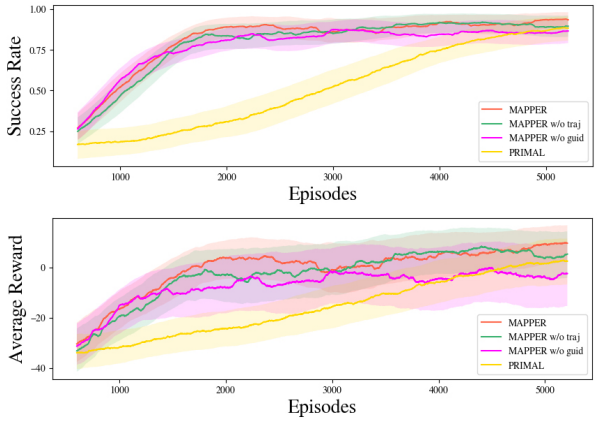

Fig. 3: Success rate and average reward comparison of variants of MAPPER and PRIMAL* algorithms

Another baseline is PRIMAL [8], a reinforcement learning-based decentralized planner. We modify the original PRIMAL to adjust to our experiment setting because the original model and observation representation do not consider non-cooperative dynamic obstacles. More specifically, we use the same observation representation and network architecture as ours, but keep the original A3C training procedures and goal-conditioned approach as PRIMAL, so we name it PRIMAL* in the rest of the paper. We also conduct experiments that remove part of features of our MAPPER method, which are removing the moving dynamic obstacles' trajectory (w/o traj) and removing the global planner guidance feature (w/o guid). We evaluate the performance of each method in terms of the **success rate** in different experiment settings.

B. Training Details

Inspired by the idea of curriculum learning [29], we divide the whole training procedure into two stages and start from easier tasks. We begin by initializing a small population of agents and dynamic obstacles, and sample goals within a certain distance to let agents learn a short-range navigation policy. Then we increase the agents and dynamic obstacles number, and sample goals in the whole map.

The training parameters are the same for MAPPER and its variants. We set off-route penalty weight $\lambda = 0.3$, the evolution rate and interval $\eta = 2, K = 50$, the discount factor $\gamma = 0.99$, and the learning rate $lr = 0.0003$. For PRIMAL*, we observe that it is sensitive to the learning rate and will not converge if we set the same learning rate as MAPPER. Therefore, we set the learning rate for PRIMAL* as 0.00005 after several experimental explorations. For the first stage, we initialize 4 agents and 10 dynamic obstacles in a 20×20 map with 7 grid goal-sample range, as shown in Fig. 2 left. For the second stage, we train models with 20 agents and 30 dynamic obstacles in a more complex 32×32 map without goal-sample limitation, as shown in Fig. 2 right.

TABLE II: Comparison of success rate over different experiment settings

Environment Setting			Success Rate				
map size	agent	dynamic obstacle	MAPPER	MAPPER w/o traj	MAPPER w/o guid	PRIMAL*	LRA*
20x20	15	10	**1.0**	0.971	0.877	0.964	0.996
20x20	35	30	**1.0**	0.961	0.836	0.980	0.999
20x20	45	30	**0.999**	0.854	0.607	0.971	0.997
60x65	70	100	**1.0**	0.256	0.516	0.352	**1.0**
60x65	130	140	**1.0**	0.473	0.221	0.404	0.992
120x130	150	40	**0.997**	0.324	0.211	0.389	0.994

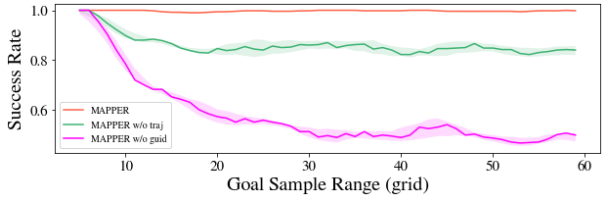

Fig. 4: Comparison of MAPPER and its variants with different goal sample range.

C. Results

The training figures for the first stage are shown in Fig 3. For the second stage training, we find that MAPPER without dynamic obstacle trajectory (MAPPER w/o traj) and MAPPER without global planner guidance (MAPPER w/o guid) can hardly converge if we sample goals from the whole map, so we limit the goal sample range to 15 grids. For PRIMAL*, the proper learning rate depends on agents number because of its centralized training nature, so we keep the agents size and learning rate as in the first stage. Since the training settings are different for second stage, the training figures are not presented in Fig 3. But from the first stage training plot, we can see MAPPER has the most stable performance (smallest variance) and fastest convergence. The final average reward and success rate of MAPPER are also superior to the other methods in comparison.

To demonstrate the effectiveness of our observation representation, we evaluate trained models in a 65×65 size simulation environment with 10 agents, 10 dynamic obstacles, and different goal sample range. The success rate when we increase the goal range is shown in Fig. 4. We can see the performance of other variants of MAPPER is sub-optimal, while MAPPER agents will not be influenced by the goal range. Specifically, if we remove the global planner guidance feature, the agent's performance declines a lot when the distance to goal is increased, which means decomposing the long-range navigation task to several easier waypoint-conditioned tasks is necessary. Though removing dynamic obstacle trajectory feature will not be influenced a lot when the goal range is changed, however, it shows worse capability to handle interactions with dynamic obstacles in a large environment.

We also evaluate the trained models as well as LRA* in various environment settings without goal sample range limitation to see their generalization capability. The performance is shown in Table II. Note that LRA* needs to access all the agents' (not dynamic obstacles) intention information and resolve conflicts before taking actions, while MAPPER only needs local observations. We observe that in simple tasks where only a few moving obstacles are around the MAPPER agent, the agent will behave similar to following the reference path from the global planner. However, when the dynamic obstacle density is increased and the reference path is blocked, MAPPER agent performs aggressively to get out of surrounding obstacles and then moves towards its goal. We can see the success rate for MAPPER is the highest and is consistently above 0.99 in various experiment settings.

The MAPPER variant without dynamic obstacle trajectory works well when there are few dynamic obstacles but performs poorly when the complexity of the environment increases. It can be seen that the waypoints guidance is an important aspect of the MAPPER algorithm and the variant without waypoints guidance has low success rate even in a 20×20 grid world with 15 agents and 10 dynamic obstacles.

VI. CONCLUSION

This paper proposes a decentralized partially observable multi-agent path planning with evolutionary reinforcement learning (MAPPER) method to learn an effective local planning policy in mixed dynamic environments. We model dynamic obstacle's behavior with an image-based representation and decompose the long-range navigation task into many easier waypoint-conditioned sub-tasks. Furthermore, we propose a stable evolutionary training approach that could be easily scaled to large and complex environments while maintaining good convergence property compared with centralized training methods. The experiment result shows that MAPPER outperforms traditional method LRA* and learning-based method PRIMAL* in terms of success rate among various experiment settings. However, MAPPER may still collide with other agents or dynamic obstacles in complex environments in order to reach the goal. So the future direction would be to investigate safety-critical learning-based planning methods.

ACKNOWLEDGMENT

The authors acknowledge the support from the Manufacturing Futures Initiative at Carnegie Mellon University made possible by the Richard King Mellon Foundation.

REFERENCES

[1] 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.

[2] 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.

[3] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[4] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[5] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[6] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[7] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1190–1197.

[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] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[10] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.

[11] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.

[12] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 220–15 231.

[13] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[14] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[15] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[16] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[17] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[18] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017, pp. 1527–1533.

[19] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[21] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 183–221.

[22] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[23] A. Pierson, C.-I. Vasile, A. Gandhi, W. Schwarting, S. Karaman, and D. Rus, "Dynamic risk density for autonomous navigation in cluttered environments without object detection," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5807–5814.

[24] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing, and T. Braunl, "Cooperative multi-robot navigation, exploration, mapping and object detection with ros," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 1083–1088.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[26] R. Lowe, Y. 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.

[27] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[28] D. Silver, "Cooperative pathfinding," *AIIDE*, no. 1, p. 117–122, 2005.

[29] S. Forestier, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *arXiv preprint arXiv:1708.02190*, 2017.