SACHA: Soft Actor-Critic with Heuristic-Based Attention for Partially Observable Multi-Agent Path Finding

Qiushi Lin and Hang Ma

Abstract—**Multi-Agent Path Finding (MAPF) is a crucial component for many large-scale robotic systems, where agents must plan their collision-free paths to their given goal positions. Recently, multi-agent reinforcement learning has been introduced to solve the partially observable variant of MAPF by learning a decentralized single-agent policy in a centralized fashion based on each agent's partial observation. However, existing learning-based methods are ineffective in achieving complex multi-agent cooperation, especially in congested environments, due to the non-stationarity of this setting. To tackle this challenge, we propose a multi-agent actor-critic method called Soft Actor-Critic with Heuristic-Based Attention (SACHA), which employs novel heuristic-based attention mechanisms for both the actors and critics to encourage cooperation among agents. SACHA learns a neural network for each agent to selectively pay attention to the shortest path heuristic guidance from multiple agents within its field of view, thereby allowing for more scalable learning of cooperation. SACHA also extends the existing multi-agent actor-critic framework by introducing a novel critic centered on each agent to approximate Q-values. Compared to existing methods that use a fully observable critic, our agent-centered multi-agent actor-critic method results in more impartial credit assignment and better generalizability of the learned policy to MAPF instances with varying numbers of agents and types of environments. We also implement SACHA(C), which embeds a communication module in the agent's policy network to enable information exchange among agents. We evaluate both SACHA and SACHA(C) on a variety of MAPF instances and demonstrate decent improvements over several state-of-the-art learning-based MAPF methods with respect to success rate and solution quality.**

Index Terms—**Path Planning for Multiple Mobile Robots or Agents; Reinforcement Learning; Deep Learning Methods**

I. Introduction

RECENT studies have demonstrated the practical success of Multi-Agent Path Finding (MAPF) [1] in various domains, such as warehouse and office robots [2], [3], autonomous aircraft-towing vehicles [4], and other multi-robot systems [5]. MAPF aims to plan collision-free paths for a set of agents from their start positions to their goal positions in a shared environment while minimizing the sum of their completion times (i.e., the arrival times at their goal positions).

Although MAPF is NP-hard to solve optimally [6], [7], the AI community has developed many optimal and bounded-suboptimal MAPF planners for fully observable environments, where a centralized planner has complete information of the environment to plan joint paths for agents. These planners do not apply to agents with limited sensing capabilities and do not scale well to a large number of agents, as the complexity of coordinating the joint paths of agents grows exponentially with the number of agents in the systems. Learning-based methods with centralized training and decentralized execution have been proposed to develop scalable and generalizable learning-based MAPF methods for the partially observable setting. In this setting, each agent receives a partial observation of its surroundings. Learning-based MAPF methods aim to train a decentralized homogeneous policy that each agent will follow based on its local observation during execution. This policy can be distributed to any number of agents in any environment, as the dimension of the single-agent observation space depends only on the FOV size in the partially observable setting. However, the non-stationarity of environments from the perspective of any single agent poses a significant challenge for learning-based MAPF methods. The transitions of the global state are affected by the individual actions of other agents towards their local interests. Moreover, goal-oriented reinforcement learning with single-agent rewards makes the training process unstable and time-consuming, further incentivizing the selfishness of each agent that prioritizes its goal over collaborating with others. This could hinder coordination and teamwork among agents, negatively affecting overall performance.

To address the challenges posed by solving MAPF in the partially observable setting, we propose SOFT ACTOR-CRITIC WITH HEURISTIC-BASED ATTENTION (SACHA), a novel approach for partially observable MAPF that leverages heuristic guidance through attention mechanisms to learn cooperation. SACHA builds upon the multi-agent actor-critic framework and, along with its communication-based alternative, SACHA(C), aims to learn a decentralized homogeneous policy that can be generalized to any number of agents in any arbitrary partially observable MAPF environment. To achieve this, we first allow each agent to access the goal-oriented heuristic guidance of multiple agents in the form of the shortest path distances to each of the goals, which can be computed efficiently before execution. We then employ a self-attention module in the policy network for each agent to locally select relevant information from the guidance and take actions towards better cooperation among agents.

Manuscript received: February 9, 2023; Revised May 11, 2023; Accepted June 14, 2023.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the NSERC under grant number RGPIN2020-06540 and a CFI JELF award.

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

Digital Object Identifier (DOI): see top of this page.

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

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

II. Problem Formulation

In this section, we first present the standard formulation of MAPF and then dive into its partially observable variant.

A. Standard MAPF Formulation

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

B. Partially Observable Environments

In this paper, we consider a more practical scenario where agents have only partial observation of the environment but still aim to fully cooperate to minimize the average

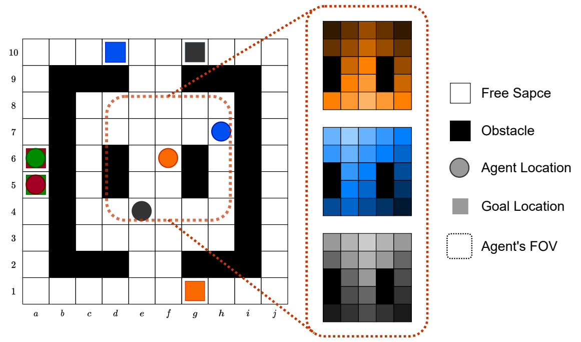

Fig. 1: Illustration of a partially observable MAPF instance and the multi-agent heuristic maps for the orange agent. Each agent's current (circle) and goal (square) positions have the same color. The orange agent's 5×5 FOV contains 3 agents, including itself. Therefore, the policy network of the orange agent will utilize 3 corresponding heuristic maps within the same 5×5 area. A darker shade of color represents a greater distance to the goal.

completion time. We model this partially observable variant of MAPF as a decentralized partially observable Markov Decision Process (Dec-POMDP) [11], defined as a 7-tuple $\langle \mathcal{S}, \mathcal{A}_i, P, \Omega_i, O, R, \gamma \rangle$. $\mathcal{S}$ is the set of global states. $\mathcal{A}_i$ is the set of actions for agent i, and $\mathcal{A} = \prod_{i=1}^{M} \mathcal{A}_i$ is the joint action space. Ω_i is the observation space of agent i, and $\Omega = \prod_{i=1}^{M} \Omega_i$ is the joint observation space. $\mathcal{O} : \mathcal{A} \times \mathcal{S} \to \Omega$ is the observation function, denoting the probability $P(o|a, s)$, whereas $\mathcal{P} : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ is the state-transition function for the environment, representing the probability $P(s'|a, s)$, where $o \in \Omega$, $a \in \mathcal{A}$, and $s, s' \in \mathcal{S}$. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor.

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

One of the key challenges for decentralized planners with limited access to global information is the occurrence of deadlocks and livelocks. Similar to time-windowed MAPF planners [12], these issues arise due to the limited planning horizon of agents, either in time or space, that prevents them from reaching their goals. For instance, consider Fig. 1, where the red agent in $a5$ and the green agent in $a6$ are heading towards their respective goals $a6$ and $a5$. An optimal solution may require the red agent to move north and terminally stay at its goal while the green agent moves north and takes a detour

TABLE I: Comparison of learning-based methods for partially observable MAPF

Method	Learning Framework	Communication	Single-Agent Guidance	Cooperative Guidance
PRIMAL [8]	A3C (RL) + Behavior Cloning (IL)	Inapplicable	Goal Direction	Goal Directions of Neighbouring Agents
DHC [9]	IQL	Required	Shortest Path Distances	–
DCC [10]	IQL	Required	Shortest Path Distances	–
SACHA (Ours)	Multi-Agent Soft Actor-Critic	Optional	Shortest Path Distances	Shortest Path Distances of Neighbouring Agents

since its direct path is blocked by the red agent. However, after a few moves, the green agent will no longer observe the blocking red agent and end up wiggling between $a9$ and $a10$ indefinitely. Symmetrically, if the green agent moves south and terminally stays at its goal, the red agent will eventually wiggle between $a1$ and $a2$.

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

III. Related Work

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

A. Learning-Based MAPF Methods

Recent learning-based MAPF methods [8], [13], [9], [10] have been proposed to solve MAPF in a partially observable setting. These methods aim to learn a decentralized policy that can be generalized to different MAPF instances. While centralized MAPF planners require full observation of the environment and must plan paths from scratch for each instance, the well-trained model can be applied to MAPF instances with any number of agents and environment size, without increasing the time complexity.

The most straightforward approach for tackling partial observability is to treat other agents as part of the environment and let each agent learn its policy independently, as in Independent Q-Learning (IQL) [14]. However, this approach results in non-cooperative behavior among the agents, and its training is not guaranteed to converge due to interference between the policies of different agents. State-of-the-art MAPF methods [9], [10] enhance IQL with a communication mechanism to promote cooperation between agents. Other MAPF methods [8], [13] use the actor-critic framework, with guidance from an expert demonstration. PRIMAL [8] combines on-policy asynchronous advantage actor-critic (A3C) [15] with behavior cloning from an expert demonstration generated by a centralized MAPF planner [16]. However, the centralized MAPF planner requires solving numerous MAPF instances, which slows down the training process. DHC [9] and DCC [10] have shown that using single-agent shortest path distances as heuristic guidance for goal-oriented learning of each agent is more effective than following a specific reference path in a multi-agent cooperative setting.

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

B. Cooperative Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is a well-established framework for coordinating multiple agents in a shared environment. A rich literature [17], [18], [19] on cooperative MARL has been dedicated to coordinating agents that work towards a common objective and take actions that benefit all agents as a whole. To deal with the nonstationarity of the environment, most existing actor-critic methods use one or more fully centralized critics that observe the entire environment. For example, Multi-Agent Deep Deterministic Policy Gradient [20] trains each actor with only its local observation using the DDPG algorithm [21], while its corresponding fully centralized critic can access the observations and actions of all agents. Instead of using multiple fully centralized critics, Counterfactual Multi-Agent Policy Gradient [22] uses only one fully centralized critic that learns to assign credit to agents and estimate Q-functions for all agents based on a counterfactual baseline that marginalizes out the action of each individual agent. However, it becomes increasingly difficult to perform such credit assignments for cases with large numbers of agents. Therefore, Multiple Actor Attention-Critic[23] deploys an attention mechanism for the fully centralized critic to selectively pay attention to relevant information from all agents. SACHA also uses a similar attention mechanism but differs from existing actor-critic methods by using a novel homogeneous agent-centered critic that only takes the local information from each agent as input for generalizability to different MAPF instances instead of a fully centralized critic specific to only one MAPF instance.

IV. SACHA

We now provide a detailed description of the main components of SACHA. First, we describe how we use the shortest path distances of multiple agents as the cooperative guidance for each individual agent. Next, we explain how we integrate the shortest path distances into the commonly applied reward design. We also elaborate on our new model design, which includes attention mechanisms applied to both the actors and the critic and an optional communication module. Lastly, we discuss SACHA in the context of the multi-agent actor-critic learning framework.

TABLE II: Individual reward function from DHC [9].

Action	Reward
Move (up / down / left / right)	-0.075
Wait (on goal, away goal)	0, -0.075
Collision (obstacles or agents)	-0.5
Reaching Goal	3

A. Multi-Agent Shortest Path Distance Heuristic Maps

Empirical studies [10] have shown that shortest path distances from all vertices to each agent's goal vertex can greatly benefit goal-oriented learning for the agent in partially observable environments. SACHA utilizes multi-agent shortest path distances to guide not only the achievement of single-agent goals but also better cooperation between agents. Specifically, a backward uniform-cost search is run from each agent's goal vertex to all vertices in the given graph to generate the shortest path heuristic maps for the agent. The heuristic maps for all agents can be calculated offline once the graph and all goal vertices are given. They remain unchanged for the same MAPF instance during training and can be efficiently generated in advance for a new MAPF instance during execution. The time complexity for calculating the distances for M agents on graph $G = (V, E)$ is $\mathcal{O}(M|E|\log|V|)$. Many search-based and some recent learning-based MAPF methods [24], [9], [10] also use heuristic maps of shortest path distance, but only as single-agent guidance. SACHA gives each agent access to not only its heuristic map but also the heuristic maps of other agents within its FOV, which enables better cooperation. Fig. 1 visualizes the heuristic maps that the orange agent has access to at the current time step. Since there are three agents within its FOV, three corresponding heuristic maps are input to the agent's policy network that we will describe later.

B. Reward Design with Heuristic Maps

We design the reward function for each individual agent based on its heuristic map. We start with the individual reward function of DHC [9] as shown in Table II. It follows the intuition that an agent is punished slightly for each time step before arriving at the goal, thereby encouraging it to reach its goal as quickly as possible. To improve the success rate of solving the global MAPF task, each agent is punished to a greater extent each time it collides with another agent or an obstacle. The agent receives a positive reward only when it arrives at its goal. Most existing learning-based MAPF methods follow the same design idea for their reward functions. However, the goal-conditioned reward in this reward design makes the training unstable and difficult to converge, especially for long-horizon tasks such as MAPF. Also, since an agent which stays further away from its goal generally has a larger potential to collide with other agents, the move actions for it should not be rewarded the same as for ones that are closer to their goals. Therefore, motivated by Heuristic-Guided Reinforcement Learning (HuRL) [25], we reshape the reward function for each agent with an additional heuristic term. Assume we have a transition tuple, (s, a, r, s'), we reshape the reward as:

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

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

C. Model Design with Attention Mechanisms

We propose a novel model architecture based on the multi-agent actor-critic framework. Our model aims to achieve generalization across different instances by restricting the actor and the critic to operate only within the observation of each agent. At each time step t, we define an undirected observation graph $G_t = (V, E_t)$, where V is the set of all agents and each edge in E_t indicates that the corresponding agents can observe each other. The time-varying graph G_t captures the dynamic correlation of agents in partially observable environments. We denote the subgroup centered on agent i as $\{i \cup \mathcal{N}_t(i)\}$, where $\mathcal{N}_t(i)$ is the set of the nearest $K-1$ neighbors of node i inside its FOV. The observation of each agent i consists of a set of K feature maps $\mathcal{F}_i = \{\mathcal{F}_i^j\}_{j \in i \cup \mathcal{N}_t(i)}$ where $\mathcal{F}_i^j \in \mathbb{R}^{\mathcal{L} \times \mathcal{L} \times 3}$. Each feature map in the set corresponds to an agent in the subgroup of agent i and contains three channels: (1) a binary matrix that identifies the obstacles and the free space, (2) a binary matrix that marks the positions of agents, and (3) a heuristic channel that shows the normalized shortest path distances for each empty cell. K is set to 3 in our experiments.

We present the learning framework of SACHA in Figure 2. Given the observation features, the policy network starts with the observation encoders with shared parameters. The encoders consist of several convolutional layers followed by a GRU [25] memory unit. The output set of K encoding is input into the Multi-Head Attention (MHA) [26] module that learns the interaction between agent i and its subgroup members by selectively attending to relevant information. The MHA module outputs a set of features the sum of which is used for the observation representation, denoted as o_i. It then will be passed to decoders to generate the corresponding action vector $a_i \in \mathbb{R}^5$. Each element in a_i represents the probability of choosing one of the five discrete actions $\{up, down, left, right, stay\}$.

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

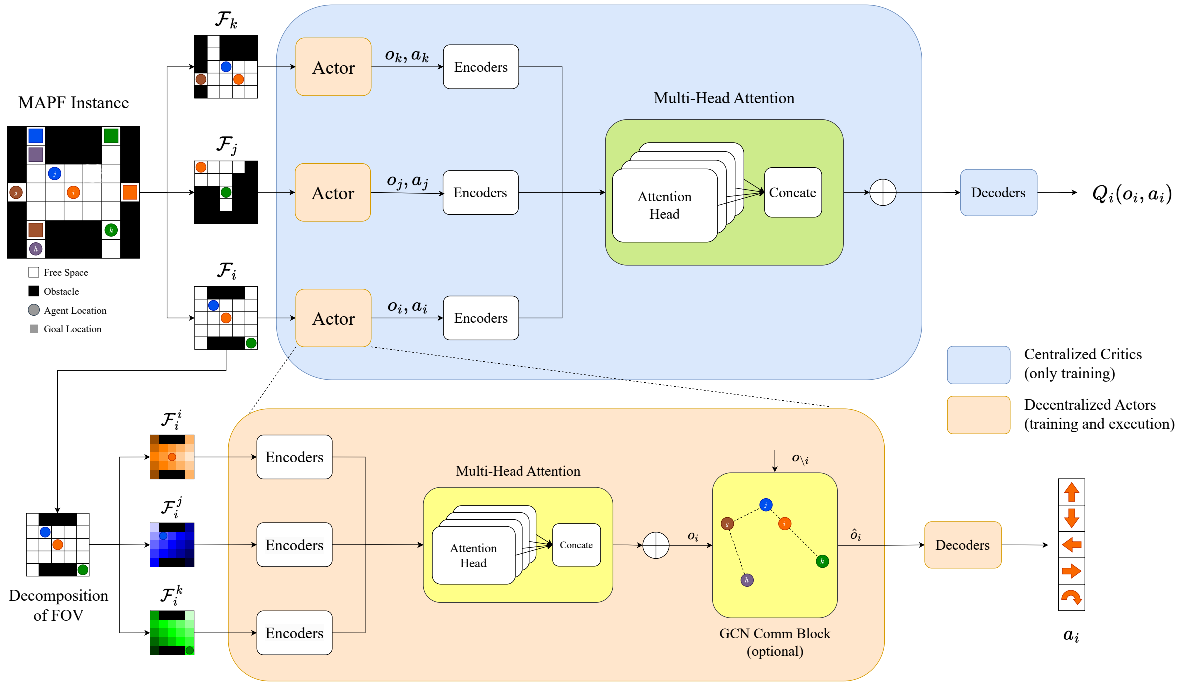

Fig. 2: Model design of SACHA and SACHA(C). The framework consists of the actors (red) and the critics (blue). The bottom part demonstrates agent i's policy network. Within its 5×5 FOV, there are three agents inside and hence three heuristic maps are fed into its network. After going through a set of encoders, three output features will then be processed by the attention block, picking out the relevant information required for agent i to take its action. The upper part shows how the critic assigns the credit for agent i based on the local observations and the corresponding actions from agents in its subgroup via a similar attention mechanism.

D. Optional Communication Module

Furthermore, we propose a communication-based variant of our method, named SACHA(C). To encourage better cooperation, our method should be able to take advantage of communication when it is allowed. We add an optional communication module after the multi-head attention block, as shown in Fig. 2. We first gather all observation representation $\{o_i\}_{i=1}^{M}$ as the initialization ,$H^{(0)}$, and then feed it into a multi-layer Graph Convolution Network [27] (GCN). Recalled that M is the number of agents. Let A_t be the adjacency matrix of G_t and $\tilde{A}_t = A_t + I_M$. Define $\tilde{D}_t$ as a diagonal matrix where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The output of the $(l+1)$-th layer would be:

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

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

E. Soft Actor-Critic with Multi-Agent Advantage Function

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

$$\begin{aligned}\delta_i &= Q_i^{\psi}(o_i, a_i) - \tilde{r}_i \\ &\quad - \gamma \mathbb{E}_{a_i' \sim \pi_{\bar{\theta}}(o_i')}[Q_i^{\bar{\psi}}(o_i', a_i') - \alpha \log(\pi_{\bar{\theta}}(a_i' | o_i')))]\end{aligned} \tag{3}$$

where α is the temperature parameter that decides the weight of the entropy term in the soft actor-critic framework [28]. SACHA runs the critic network through every agent-centered subgroup and updates it by minimizing the mean square error loss function:

$$L_Q(\psi) = \mathbb{E}_{e \sim D} \sum_{i=1}^{M(e)} \frac{\delta_i^2}{M(e)}, \tag{4}$$

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

$$\begin{aligned}\nabla_{\theta_i} J(\theta) &= \mathbb{E}_{o_i \sim D, a_i \sim \pi_{\theta_i}(o_i)}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i | o_i)) \\ &\quad (Q_i^{\psi}(o_i, a_i) - b(o_i, a_{\setminus i}) - \alpha \log(\pi_{\theta_i}(a_i | o_i)))]\end{aligned} \tag{5}$$

where $\nabla_\theta \log(\pi_\theta(a_i|o_i))$ is the score function and $Q_i^\psi(o_i, a_i) - b(o_i, a_{\setminus i})$ is the multi-agent advantage function. Inspired by COMA [22], SACHA adopts the counterfactual baseline in a discrete action space as follows, which marginalizes out the specific action of agent i:

$$b(o, a_{\setminus i}) = \sum_{a'_i \in \mathcal{A}_i} \pi(a'_i|o_i) Q_i^\psi(o_i, (a'_i, a_{\setminus i})), \tag{6}$$

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

V. Analysis

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

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim D, a \sim \pi_\theta}[\nabla_\theta \log(\pi_\theta(a|s)) Q(s, a)], \tag{7}$$

where s and a denote the joint state and joint action, respectively. Now we show that this gradient is equivalent to Eq. (5). We omit the entropy term here, but the proof can be easily extended to include it.

Since each agent acts on its policy independently, we have $\pi_\theta(a|s) = \prod_i \pi_{\theta_i}(a_i|s)$. We represent the stationary distribution induced by π_θ as $d_\theta(s)$, meaning the probability of being in the state s by following π_θ. Following the proof in [29], we get:

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

To further the proof, we consider the following equation:

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

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

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

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

VI. Experiments

In this section, we implement our methods[1] and experimentally evaluate them with other methods on a server equipped with an Intel 2.3GHz 16-Core CPU and an NVIDIA A40 GPU.

A. Environment Setups

Training Environments: As mentioned above, our model is trained using the multi-agent actor-critic learning framework. Not only does each agent's policy network share parameters but also the critic networks applied to each subgroup of agents are homogeneous. We train our model over random grid maps of different sizes with randomly generated obstacles. The obstacle density is sampled from a triangular distribution between 0 and 0.5 with its peak value at 0.33. To fairly compare with other decentralized MAPF planners, our agent's policy network exclusively has 9×9 square-shaped FOV, the same as DHC and DCC, regardless of the environment size. Inspired by the curriculum learning [30], we design a training pipeline that starts with only 2 agents on 10×10 grid maps and gradually increases the number of agents and the size of the map once the success rate reaches a certain threshold. More and more complicated tasks are constantly added to the training pool until the map size exceeds 100×100 or the number of agents exceeds 72.

Testing Environments: We test our methods over a variety of maps all from the standard benchmark [1]. We first select

[1]The code is available at https://github.com/Qiushi-Lin/SACHA.

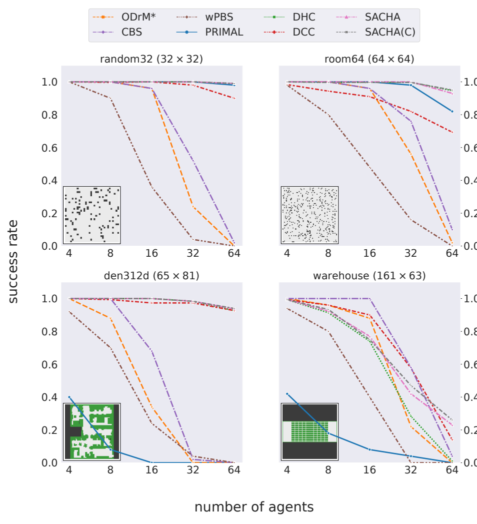

Fig. 3: Success rates for different learning-based MAPF methods on different maps.

TABLE III: Solution quality for different MAPF methods.

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

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

B. Baselines

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

Search-Based Methods: Furthermore, we also compare our method with the centralized planners. We use two optimal but computationally expensive search-based methods for comparison: Conflict-Based Search (CBS) [24] and ODrM* [31]. CBS performs a best-first search that expands the constraint tree node by adding constraints to each agent involved in every conflict, while ODrM* is a suboptimal planner based on M* that applies Operator Decomposition (OD) to split agents into independent conflict sets and thus reduce the complexity of joint planning. We also use Priority-Based Search (PBS) [32] that searches for certain agent orders that can be used in the prioritized planning, which makes this solver incomplete but efficient. To simulate centralized planning in the same partially observable setting, we use windowed PBS (wPBS) [12], which only avoids conflicts within a bounded time horizon. We set the time window length of wPBS to be equal to the caliber of the FOV to simulate the partially observable environments. We set the runtime limit of CBS and wPBS to 120 seconds and ODrM* to 20 seconds.

C. Empirical Results

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

Fig. 3 shows the success rate of our methods compared with all other baselines over four different MAPF maps. Within the time limit, all decentralized planners have a remarkable advantage in success rate. Even by including the precomputing time of shortest path heuristics, decentralized planners find solutions much faster than centralized ones. Among decentralized planners, PRIMAL tends to result in solutions with the worst quality, especially in those two game maps, which indicates that learning from the expert data cannot be easily generalized to instances with different numbers of agents and on maps with different structures. DHC and DCC have their advantages over PRIMAL by allowing the shortest path heuristics and the communication mechanism, although our methods can both outperform them in most cases. The advantages are more obvious over maps with higher obstacle density and more agents where more cooperation is demanded.

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

As reported in [9], DHC always has better performance than its alternative without the communication unit. Hence, it shows that our method can serve the non-communicating scenarios when SACHA can outperform DHC and thus DHC without communication. Besides, DHC can essentially be viewed as training our communication-based model without the attention block via the independent Q-learning. Therefore, the fact that SACHA(C) has greater performance than DHC demonstrates the strength of the heuristic-based attention mechanism and the multi-agent learning framework. Overall, our methods have a better chance to solve given MAPF instances, with and without communication, and among those solved instances, they result in solutions with better quality.

VII. Conclusion

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

References

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

[2] 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.

[3] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *IJCAI*, 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 *AAAI Workshops*, 2016.

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

[6] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *AAAI*, 2013.

[7] J. Banfi, N. Basilico, and F. Amigoni, "Intractability of time-optimal multirobot path planning on 2d grid graphs with holes," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1941–1947, 2017.

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

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

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

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

[12] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *AAAI*, 2021, pp. 11 272–11 281.

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

[14] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *ICML*, 1993, pp. 330–337.

[15] 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 *ICML*, 2016, pp. 1928–1937.

[16] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *IROS*. IEEE, 2011, pp. 3260–3267.

[17] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[18] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers*. Springer, 2017, pp. 66–83.

[19] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Applied Intelligence*, pp. 1–46, 2022.

[20] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *NeurIPS*, vol. 30, 2017.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[22] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI*, 2018.

[23] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *ICML*, 2019, pp. 2961–2970.

[24] 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.

[25] C.-A. Cheng, A. Kolobov, and A. Swaminathan, "Heuristic-guided reinforcement learning," *NeurIPS*, pp. 13 550–13 563, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

[27] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ArXiv*, vol. abs/1609.02907, 2016.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*. PMLR, 2018, pp. 1861–1870.

[29] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *NeurIPS*, 1999.

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

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

[32] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *AAAI*, 2019, pp. 7643–7650.