http://justc.ustc.edu.cn Received: March 11, 2022; Accepted: July 04, 2022

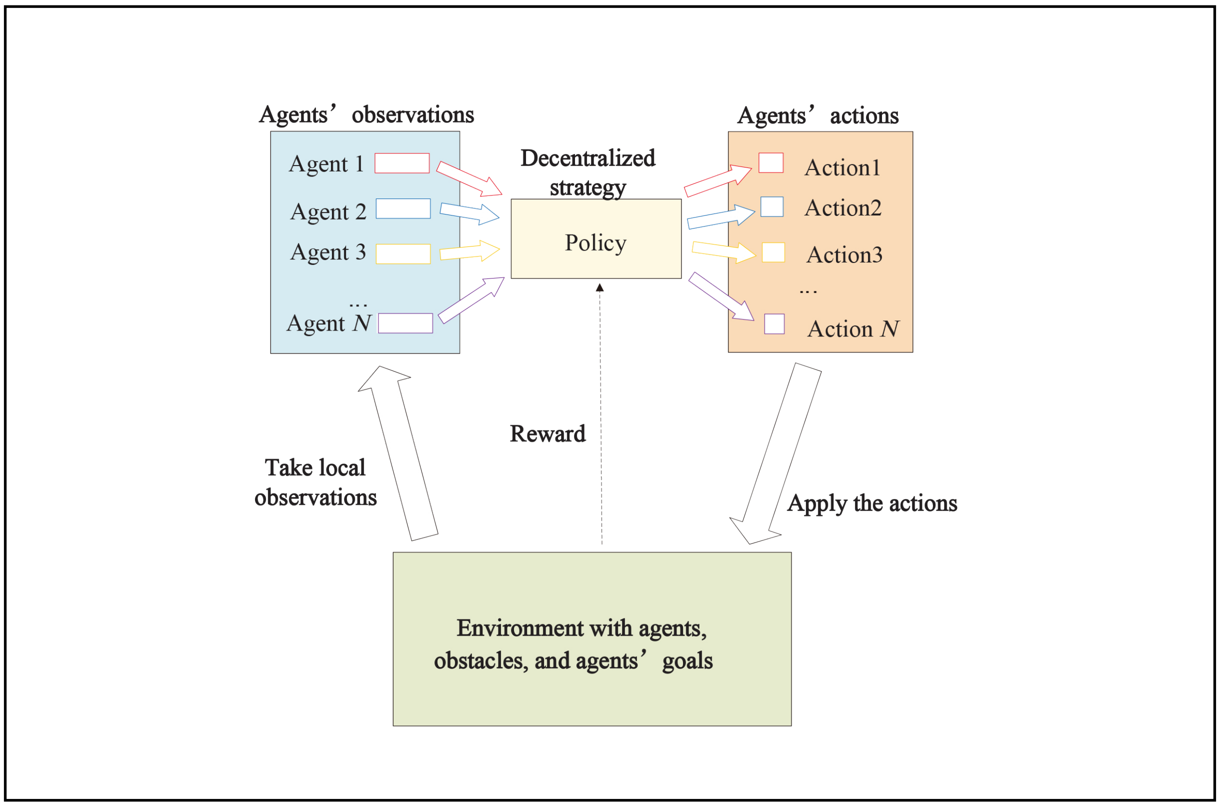
# Learning attention-based strategies to cooperate for multi-agent path finding

Jinchao Ma, and Defu Lian ✉

*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*

✉Correspondence: Defu Lian, E-mail: liandefu@ustc.edu.cn

## Graphical abstract



*To solve the problem of multi-agent path finding (MAPF), a new deep reinforcement learning model with local attention cooperation is proposed in this work.*

## Public summary

■ We build the local observation encoder by using residual attention CNN to extract local observations and use the transformer architecture to build an interaction layer to combine local observations of agents.

■ To overcome the deficiency of the success rate, we also designed a new evaluation index, namely the extra time rate (ETR).

■ The experimental results show that our model is superior to most of the previous models in terms of success rate and extra time rate.

Article

http://justc.ustc.edu.cn

Received: March 11, 2022; Accepted: July 04, 2022

# Learning attention-based strategies to cooperate for multi-agent path finding

Jinchao Ma, and Defu Lian

*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*

Correspondence: Defu Lian, E-mail: liandefu@ustc.edu.cn

Cite This: *JUSTC*, **2023**, 53(4): 0404 (12pp)     **Read Online**

**Abstract:** Multi-agent path finding (MAPF) is a challenging multi-agent systems problem where all agents are required to effectively reach their goals concurrently with not colliding with each other and avoiding obstacles. In MAPF, it is a challenge to effectively express the observation of agents, utilize historical information, and effectively communicate with neighbor agents. To tackle these issues, in this work, we proposed a well-designed model that utilizes the local states of nearby agents and outputs an optimal action for each agent to execute. We build the local observation encoder by using residual attention CNN to extract local observations and use the Transformer architecture to build an interaction layer to combine local observations of agents. With the purpose of overcoming the deficiency of success rate, we also designed a new evaluation index, namely extra time rate (ETR). The experimental results show that our model is superior to most previous models in terms of success rate and ETR. In addition, we also completed the ablation study on the model, and the effectiveness of each component of the model was proved.

**Keywords:** multi-agent path finding (MAPF); reinforcement learning; decentralized planning; attention mechanism

**CLC number:** TP181     **Document code:** A

## 1 Introduction

Multi-agent path finding (MAPF) is a significant problem in multi-agent planning problems, whose goal is to effectively plan paths for multiple agents with the constraint that agents need to avoid colliding with obstacles and other agents[1]. MAPF is a common problem in many practical scenarios. For example, MAPF is deployed in automated warehouses[2, 3], airplane taxiing[4, 5], automated guided vehicles (AGVs)[6, 7] and so on. Indeed, MAPF is full of challenges because solving for the optimal method is NP-hard[8], and when the size of the environment, the number of agents, and the density of obstacles increase sharply, a large number of potential path conflicts may occur.

Generally, there are two major categories in the multi-agent system: centralized methods and decentralized methods. Centralized methods apply a single learner to discover joint solutions (team learning), while decentralized methods use multiple simultaneous learners, usually one for each agent (concurrent learning)[9]. The same classification also exists on MAPF issues[1]. In the centralized MAPF, all agents' partial observations are known and collected to generate collision-free paths for them, while in decentralized MAPF, it is not necessary to know all the states of all agents to determine each agent, and each agent makes decisions independently according to its own local observation. With the growth of the number of agents, world scale, and obstacle density, computational complexity will be an important concern, especially for centralized methods. Therefore, in this paper, we focus on decentralized methods, where effective and efficient communications between agents are crucial. Moreover, the information for communication also needs to be carefully coded, and it needs to consider not only the effective coding of local observations but also the effective combination of historical state information. Therefore, we use convolutional neural networks (CNN) with residual structures to encode local observations and gated recurrent unit (GRU) cells to integrate historical information.

In the past few decades, decentralized solutions to this problem have attracted extensive interest. In traditional methods, the reaction method is an intuitive and widely adopted benchmark. Take local repair A* (LRA*)[10], for example, in which each agent searches for its own path to its own goal using the A* algorithm, ignoring all other agents except for its current neighbors. Obviously, there will be many conflicts, so dealing with conflicts effectively is the key work of reaction. Whenever a collision is about to occur, the agent needs to avoid this situation by recalculating the remainder of its route and considering the occupied grids. In addition to reaction-based methods, there are also conflict-based methods (CBS[11], MA-CBS[12], ECBS[13], and ICBS[14]). In general, CBS[11] is a two-level search algorithm. While CBS calculates the path of each agent independently at the low level, and at the high level it detects conflicts between agent pairs and solves the conflict by splitting the current solution into two related subproblems, each of which involves replanning a single agent. By dividing the subproblem into two subproblems to solve the conflict recursively, the search tree is implicitly defined. Advanced search searches this tree with best-first search and terminates when the conflict-free leaf is extended.

In recent years, there has been some seminal work using deep architectures to automatically find optimal or suboptimal solutions for planning problems. These approaches vary from supervised learning to deep reinforcement learning, and their structures contain CNN[15, 16], LSTM[17], and GNN[18, 19]. In the previous approaches, some works (for example, PRIMAL[15], PRIMAL$_2$[16]) did not think communication among agents is necessary and did not make any effort to design communication structures, while some works (for example, LSTM[17], GNN[18], and GAT[19]) thought the communication was very important and they took all the effort to build communication structures for agents to share their state information.

More importantly, these previous works did not distinguish the importance of local information, so we need to distinguish the importance of local information to extract more impressive features. Inspired by MAAC (multi-actor-attention-aritic), which uses an attention mechanism in multi-agent reinforcement learning to select relevant information for each agent at every time step, we use an attention mechanism to share neighbors' information for each agent. Reinforcement learning methods also face the problem of solving long-horizon tasks with sparse rewards[20], especially when the environment is very large, and the training process will be inefficient. To speed up the training process in reinforcement learning, some of the previous works also used imitation learning for the cold start dilemma in reinforcement learning[15, 16], while curriculum learning for the environment setting also made sense where the easy tasks took some inspiration for hard tasks and humans explored complex tasks from easy step by step[16].

The major contributions of this paper are as follow:

( i ) To solve the problem of multi-agent path finding (MAPF), a new deep reinforcement learning model with local attention cooperation , called local attention cooperation reinforcement learning (LACRL), is proposed in this work.

( ii ) We build the local observation encoder by using residual attention CNN to extract local observations and use the transformer architecture to build an interaction layer to combine local observations of agents.

(iii) To overcome the deficiency of the success rate, we also designed a new evaluation index, namely the extra time rate (ETR). The experimental results show that our model is superior to most of the previous models in terms of success rate and extra time rate.

## 2 Related works

**Classical path planning methods.** Generally, there are two major categories in MAPF approaches: coupled (centralized) and decoupled (decentralized). There will be a catastrophe for centralized approaches when the size of the world and the number of agents grow sharply, and the performance of centralized approaches will struggle because of the tremendous search space. Almost all traditional methods tend to use centralized or coupled methods, which use the complete information of all agents and the world environment to plan the path globally. Among the traditional methods, the most famous and well-used decoupled methods are search-based (for example, LRA* and CBS) methods. A*-based (for example, LRA* and WHCA*) algorithms rely on complete observations and use A* to calculate full paths for each agent, which could work in both centralized (CBS, WHCA, ORCA) and decentralized manner[6]. In A*-based methods, the reaction trick is intuitive and widely adopted, so there are also some works that take the reaction-based methods as a branch of approaches. Take LRA*, for example, in which each agent searches for its own path to its own goal using the A* algorithm, ignoring all other agents except for its current neighbors. Obviously, there will be many conflicts, so dealing with conflicts effectively is the key work of reaction. Whenever a collision is about to occur, the agent needs to avoid this situation by recalculating the remainder of its route and considering the occupied grid. Besides search-based methods, there are also conflict-based search and its variants (CBS, MA-CBS, ECBS, and ICBS), which make a plan for each single agent and construct a set of constraints to find the optimal or near-optimal solution without exploring the high-dimensional spaces. In conflict-based approaches, CBS is the most original and influential method, which is a two-level search algorithm. While CBS calculates the path of each agent independently at the low level, at the high level, it detects conflicts between agent pairs and solves the conflict by splitting the current solution into two related subproblems, each of that involves replanning a single agent. By dividing the subproblem into two subproblems to solve the conflict recursively, the search tree is implicitly defined. Advanced search searches this tree with the best-first search and terminates when the conflict-free leaf is extended.

In addition to search-based methods and conflict-based methods, there are also thousands of trick to solve the problem, such as direction map[21], subgraph structure[22], flow annotation replanning[23], increasing cost tress search (ICTS)[24–26], satisfiability[27–30], integer linear programming[31–33], and answer set programming[34]. In the direction map method[21], there is a direction map (DM) that stores information about the direction that agents have traveled in each portion of a map. In the planning process, agents then use this information, in which movement that runs counter to the pattern incurs additional penalties, thus encouraging delegates to move more evenly throughout the environment. In the subgraph structure method[22], they propose a new abstract form to plan more effectively, in which the key of this approach is to divide the mapping into subgraphs with known structures. These known subgraphs have entry and exit constraints, and can be compactly represented. Then, planning becomes a search in a smaller subgraph configuration space. Once an abstract plan is found, it can be quickly decomposed into correct (but possibly suboptimal) concrete plans without further search. ICTS[24–26] interweaves two search processes. The first is called advanced search, which aims to find the size of the agent's single agent plan in the optimal solution of a given MAPF problem. The second is called low-level search, which accepts a plan size vector and verifies whether there is an effective solution for a given MAPF problem. Finally, there are some methods that transform the MAPF problem into a different problem for which good solvers exist, such as satisfiability[27–30], integer linear programming[31–33], and answer set programming[34].

In fact, solving the optimality is NP-hard. Although significant progress has been made in reducing the amount of computation, the scalability of these methods in an environment with a large number of potential path conflicts is still very poor. To reduce the number of hand-tuning parameters and address sensing uncertainty, some researchers have proposed learning-based methods to solve the planning problem[20].

**Learning-based methods.** Thanks to the rapid development of deep learning technology in recent years, learning-based methods are considered to be a promising direction for solving the task of path planning. Reinforcement learning has always been a powerful tool for solving planning problems, and it successfully solves the path planning problem, in which the agent completes the task through repeated trial and error. ORCA[35] adjusts the speed, size, and direction of agents online to avoid collision. On the separately planned single agent path, recent works have focused on the obstacle avoidance method of reinforcement learning. Refs. [15, 16] proposed a hybrid learning-based method called PRIMAL for multi-agent path finding that integrated imitation learning and multi-agent reinforcement learning. In addition, there is a method of reinforcement learning combined with evolutionary thought. The approaches[15, 16] did not consider interrobot communication and thus, did not exploit the full potential of the decentralized system[20]. In other words, communication is important, especially for decentralized approaches, and it is difficult to fully estimate the intention of adjacent decision agents without communication. Ref. [18] used CNN to extract adequate features from local observations and use graph neural network (GNN) to transfer these features between all agents. Then, the model is trained offline by imitating the expert algorithm, and the model is used online for decentralized planning involving only local communication and local observation. Ref. [19] believed that vanilla GNN relies on a simple message aggregation mechanism, which hinders the agent from prioritizing important information. Therefore, they extend the previous work that uses vanilla GNNs to graph attention network (GAT). Their message-aware graph attention network (MAGAT) is based on a key-query-like mechanism that determines the relative importance of features in the messages received from various neighboring robots. Ref. [17] proposed a decentralized multi-agent collision avoidance algorithm based on deep reinforcement learning, in which introduces a strategy using LSTM that enables the algorithm to use observations of an arbitrary number of other agents instead of previous methods that have a fixed observation size.

These previous works did not distinguish the importance of local information, so we need to distinguish the importance of local information to extract more impressive features used in the last decision process. Moreover, the approaches[15–16] did not have the communication between agents and did not exploit the full environment to complete the task. This is different from Refs. [18, 19], which used a graph structure to transfer features between all agents. In contrast, only when some agents are very close do they communicate information with each other. Based on another fact, when the world is large, and the number of agents is large, it takes considerable time to build graphs and aggregate information on all agents. Therefore, in a large-scale environment, local communication will save a lot of time and respond quickly.

In terms of training methods, we do not use imitation learning[15, 16, 18, 19] and guidance path[17, 20]. We use RL framework and design all rewards to control the process of path exploration. We believe that although these training skills can speed up the efficiency of agent exploration of the environment, they also greatly affect the quality of the agent strategy. Although we don’t use the above training skills, we also make an attempt to improve the efficiency of agent exploration. Inspired by the curriculum learning used in imitation learning, we divide the training process of the model into many steps, starting from relatively simple tasks to more complex tasks.

## 3 Local attention cooperation reinforcement learning

### 3.1 Preliminary

We model MAPF under the Markov decision processes (MDPs) framework, which converts MAPF to a sequential decision-making problem in which each agent needs to take the instant action option at time $t$, with two goals: quickly reach the goal contently and make great efforts to avoid collision among agents.

**Environment.** Consider a 2-dimensional discrete environment $\boldsymbol{E} \subset \mathbb{R}^2$ with size $W \times H$ and the cell $\boldsymbol{C} \in \boldsymbol{E}$. For each cell $\boldsymbol{C}$, it has three states: free, busy (occupied by someone agent), and disable (occupied by obstacle). There are a set of $N_\mathrm{o}$ obstacles, $\boldsymbol{C}_\mathrm{o} = \{o_1, \ldots, o_{N_\mathrm{o}}\}$, where $o_i \in \boldsymbol{E}$ represents that there is the $i$th obstacle in the environment, and a set of $N_\mathrm{A}$ agents $\boldsymbol{A} = \{A^1, \ldots, A^{N_\mathrm{A}}\}$, where $A^i \in \boldsymbol{E}$ represents that the $i$th agent can travel between the free cells, and the free cells of the world can be represented by $\boldsymbol{C}_\mathrm{a} = \boldsymbol{E} \setminus \boldsymbol{O}$, $\boldsymbol{O}$ is the set of all obstacles in the world. For the $i$th agent, there is a unique start position $c_{\mathrm{s}_i}$ and a unique goal position $c_{\mathrm{g}_i}$, with the constraint that $c_{\mathrm{s}_i} \in \boldsymbol{C}$ and $c_{\mathrm{g}_i} \in \boldsymbol{C}$. Our goal is to find a decentralized planner, which plans the $i$th agent’s motion path by taking the local observation and considering the neighbor agents’ states in addition to quickly finding the scheduling path, it is also important to ensure the security of the solution, that is, try to avoid conflicts between agents.

**State’s structure.** We consider that every agent partially observes the environment $(W \times H)$, where agents only observe the grid world in a limited field of view (FOV), the radius of which is defined as $R_\mathrm{FOV}$ and 9 is actually used in our experiment. As mentioned in Ref. [15], partially observing the world is an important step towards the deployment of robots in the real world. Only in closed and small scenarios, a complete map of the environment is available (e.g., automated warehouses), and agents can be trained by using a sufficiently large FOV to complete full observation of the system. For each agent in the environment, it will have a perception of its local environment. The perceived information plays an important role in planning the motion path of the agent. In detail, at each time step $t$, the agent $A^i$ will take its local observation around it, which contains obstacle location information $L^i_o$, the location of adjacent agents $L^i_a$, the target location of current agent (if the goal in its neighboring) $L^i_g$, and the targets of other agents $L^i_{og}$. Local observations of agent $A^i$ can be formally expressed as $\boldsymbol{L}^i_{ob} = [L^i_o; L^i_a; L^i_g; L^i_{og}]$.

Additionally, assuming a fixed FOV, the strategy can be extended to any world size, which also helps to reduce the input dimension of the neural network and reduce model complexity[15]. The local state's structure of each agent can be found in Fig. 1.

**Communications.** Since agents do not have global position of others, the communication (or information sharing) between the agents are significant to complete the cooperative task. Addressing the problems of what information should be sent to whom and when is crucial to solving the task effectively[19]. We try to design a communication block, which takes the local observation of the agent and its neighbor agents as input and outputs a tensor condensing the information that will redound to do a decision-making. At time $t$, agent $A^i$ will be adjacent to some agents that can be recorded as $N^i_t$. And the local observation of the agent $A_i$ and its neighbor agents $j \in N^i_t$ can be formally expressed as $I^i = \{L^j_{ob} \mid j \in N^i_t\}$, while the output tensor can be recorded as $\boldsymbol{S}^i$.

**Action space.** We describe the MAPF problem as a sequence classification problem, which selects an optimal action from action space in each time step. And in our experiment, we consider a 4-connected grid environment, which means the agent can take 5 discrete actions in the grid world: moving a cell in one of the four basic directions or staying stationary. However, if the target mesh is already occupied by other agents or obstacles, the agent will not be able to move and will stay in their current position.

**Reward design.** The goal of MAPF is to reach the target position with the smallest stride while avoiding collision with obstacles and other agents. Therefore, there need step penalty $r_{\text{step}}$ that after an agent move a step will get a small penalty with the purpose to push the agent to quickly reach its goal. Besides, collision penalty $r_{\text{collision}}$, which should be given to the agent when it collides with obstacles or other agents, is also important and will be little bigger than step penalty. 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 Ref. [20]. Since the waiting penalty is slightly more than the moving penalty, the swing of the agent will occur frequently in our experiment. To avoid this situation, here we need to introduce swing penalty $r_{\text{swing}}$ when agents return to the position they come from last time. Finally, when the agent reach its goal, the goal-reaching rearward $r_{\text{goal}}$ will be given to the agent. The detailed values of these reward components in our experiment can be found in Table 1.

## 3.2 Our methods

In this section, we will explain how agents output actions based on inputs, and show our entire model architecture, and then explain the specific network architecture on each module. The whole model architecture is shown in Fig. 2. First, the agents locally observe their environment to get the surrounding information, and through a local attention encoder to get the expression of its local observation. Then, the agent

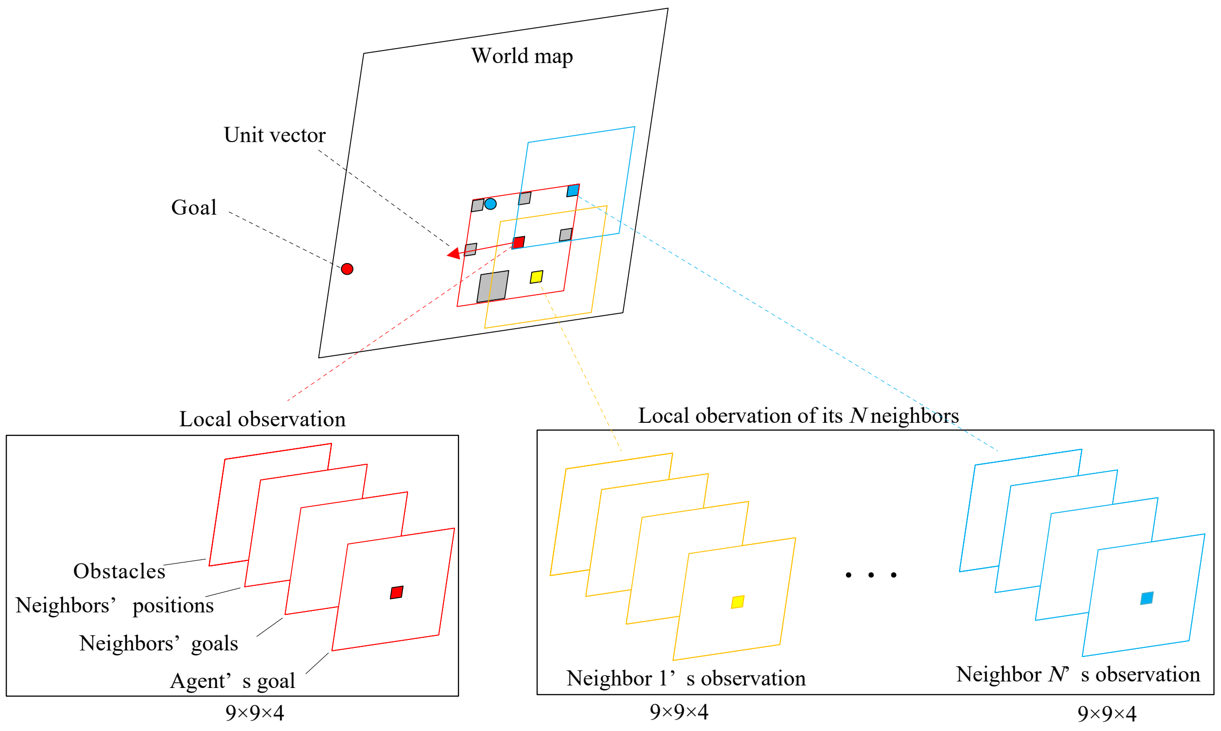


**Fig. 1.** State's structure of each agent (here, for the red agent). Agents represented as a red square are positioned in the squares, while their goals are represented as a solid circle of the same color. For the current agent (red agent), its local observation contains four channel information: positions of obstacles, positions of nearby agents, goal positions of these nearby agents, and position of its own goal (if exist in the FOV). In addition to the current (red) agent status, other nearby agents' states are also required (here, take the yellow agent and blue agent, for example).

**Table 1.** Reward design. There need step penalty $r_{\text{step}}$ to promote the agent to achieve its goal quickly. And inspired by Refs. [15, 16, 20], we subdivide the step penalty into move penalty and wait penalty. The waiting penalty (e.g., –0.5) is slightly larger than the moving penalty (e.g., –0.3), which will promote the agent to explore the environment. Since the waiting penalty is slightly more than the moving penalty, we introduce the swing penalty $r_{\text{swing}}$ to avoid the swing of the agent. In addition, collision penalty $r_{\text{collision}}$ should be given to the agent when it collides with obstacles or other agents. Finally, when the agent reach its goal, the goal-reaching reward $r_{\text{goal}}$ will be given to the agent.

| Action | Reward |
|---|---|
| Step penalty (move) | −0.3 |
| Step penalty (wait) | −0.5 |
| Collision penalty | −2.0 |
| Swing penalty | −1.0 |
| Goal-reach reward | +40.0 |

will communicate with its neighbor agents to cooperate, in which an transformer architecture is used to exchange the local observation, so the communication block can also be called the interaction layer. Finally, the tensor output from the communication block will be input to a decision block, which is considered as a learning strategy and estimates the optimal action at this time.

**Local attention encoder (LA-Encoder).** Because the motivation of this module is to extract and encode the local observations, it also can be called feature layers. In detail, at each time step $t$, the agent $A^i$ will take its local observation $\boldsymbol{L}_t^i$ around it, which be formally expressed as

$$\boldsymbol{L}_t^i = [L_{\text{o}}^i; L_{\text{a}}^i; L_{\text{g}}^i; L_{\text{og}}^i], \tag{1}$$

where $L_{\text{o}}^i$ is the obstacle location information, $L_{\text{a}}^i$ is the location of adjacent agents, $L_{\text{g}}^i$ is the target location of current agent (if the goal in its neighboring), and $L_{\text{og}}^i$ is the targets of other agents.

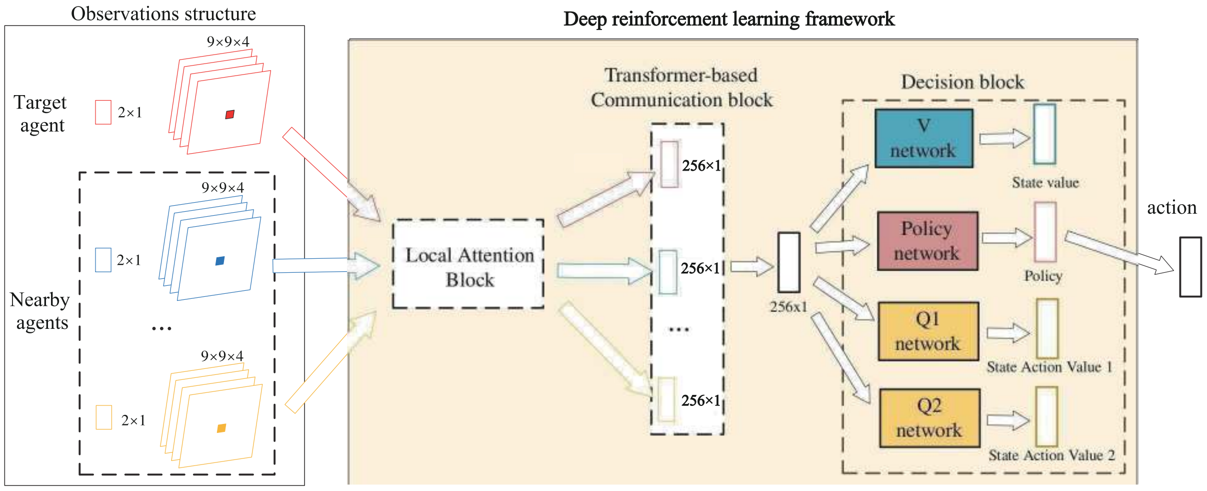


**Fig. 2.** Model overview. For the input observations, there are local observations of the target agent (red agent) that need to be planned and other agents nearby the target agent (blue agent, yellow agent, etc). For our deep reinforcement learning framework, there are three components: observation encoder, communication block, and decision block. Finally, the estimated optimal action from the policy network is taken as the output of the model.

In addition to local observation information $\boldsymbol{L}_t^i$, it is significant to introduce the local guide direction formally expressed as $\boldsymbol{v}_t^i$. The purpose is to point out the target location, especially when the grid world is too large and the target exceeds FOV. At time $t$, the cell of agent A is $\boldsymbol{C}_t^i$ and the cell of its target location is $\boldsymbol{C}_{\text{goal}}^i$, then the direction vector for local guidance can be formally expressed as $\boldsymbol{v}_t^i$. The calculation process of local guide vector $\boldsymbol{v}_t^i$ is as follows:

$$\boldsymbol{v}_t^i = \frac{C_{\text{goal}}^i - C_t^i}{|C_{\text{goal}}^i - C_t^i|}, \tag{2}$$

where $C_t^i$ is the location of agent $A^i$ at time $t$, $C_{\text{goal}}^i$ is the cell of its target location, and $|\cdot|$ means the modulus of a vector.

By using local observation information $\boldsymbol{L}_t^i$ and local guide vector $\boldsymbol{v}_t^i$, the LA-Encoder will output an intermediate expression $h_t^i$:

$$h_t^i = \text{LA-Encoder}(\boldsymbol{L}_t^i, \boldsymbol{v}_t^i). \tag{3}$$

The whole model structure of the LA-Encoder can be found in Fig. 3. The LA-Encoder is different from previous works[6, 15, 16, 18–20, 36], they only designed a deep convolutional neural network to extract features, but did not pay attention to the importance of local information, which made the expression ability of the extracted features insufficient, which affected the subsequent training difficulty of the model.

As we can see in Fig. 3, the local observation information $\boldsymbol{L}_t^i$ of agent $A^i$ will be input into a well-designed layer that uses attention mechanisms to process important information. To use the attention mechanism, the local observation $\boldsymbol{L}_t^i$ needs to be convoluted three times with convolution kernels of $1 \times 1$ to obtain $\boldsymbol{K}$ tensor, $\boldsymbol{Q}$ tensor, and $\boldsymbol{V}$ tensor. The attention calculation process, it should be noted, is carried out on each channel, because the values of each channel are in different ranges and have different meanings. Therefore, the calculated attention map is a group of attention maps on each channel, and it is used as a weight to fully extract local information on each channel. Finally, the local feature extracted by the attention mechanism will be combined with the local guidance vector $v_t^i$ to obtain the intermediate expression

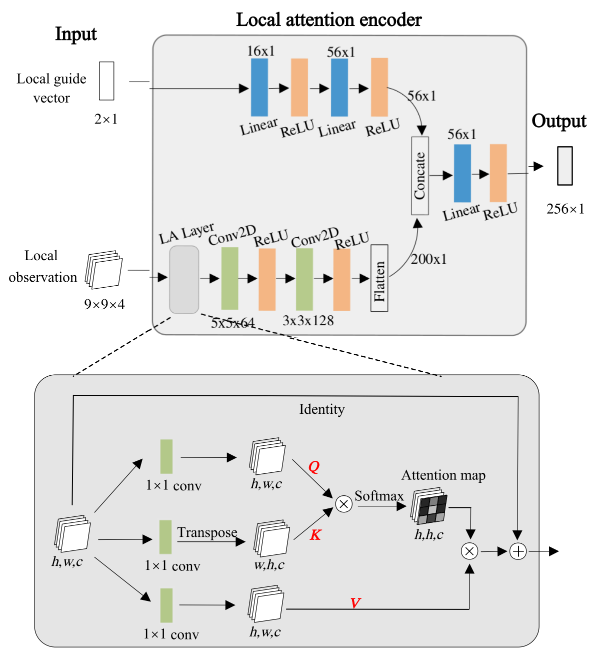


**Fig. 3.** The structure of LA-Encoder. Firstly, the local observation $L_t^i$ needs to be convoluted three times with convolution kernels of 1×1 to obtain $\boldsymbol{K}$ tensor, $\boldsymbol{Q}$ tensor, and $\boldsymbol{V}$ tensor. Then, the attention map is carried out on each channel, and it is used as a weight to fully extract local information on each channel. Finally, the local feature extracted by the attention mechanism will be combined with the local guidance vector $\boldsymbol{v}_t^i$ to obtain the intermediate expression.

$h_t^i$ containing the local state and target information, which can be applied to the subsequent planning.

**Transformer-based communication block (TC-Block).** To enable agents to perceive the state of adjacent agents, we also designed a local cooperation module based on transformer architecture, namely Transformer-based communication block (TC-Block). For a certain agent $A^i$, it will communicate with its neighbors $N^i$ and share their intermediate expressions $h^j$ $(j \in N^i)$. Then TC-Block will output the comprehensive expression $s_t^i$, which will be input into decision block to estimate optimal action at time $t$:

$$s_t^i = \text{TC-Block}(h_t^i, h_t^j),\ j \in N_t^i, \tag{4}$$

where $h_t^i$ is the intermediate expression from LA-Encoder of agent $A^i$ at time $t$, $N_t^i$ is the group of agents which is adjacent to current agent $A^i$ at time $t$.

The structure of TC-Block can be found in Fig. 4. With the purpose of combining the local states of the current agent and its neighbors, the TC-Block takes the features obtained from the LA-Encoder as input. And these encoded features of agents can be regarded as a sequence and can be used as a part of input. In addition, we also need to embed the position of each feature tensor. We use relative coordinates as position information. For example, the current agent $A^i$ is in cell $\boldsymbol{C}_t^i$, and a neighbor $A^j$ is in cell $\boldsymbol{C}_t^j$. Therefore, the relative position $\boldsymbol{C}_t^j - \boldsymbol{C}_t^i$ will be location information and serve as an input for position embedding. And inspired by Ref. [37] using transformer in image classification task, we embed the position information from $X$ coordinate and $Y$ coordinate, each with the size of half of the position embedding $D/2$. Then, based on the relative coordinate, we concatenate the $X$ and $Y$ embedding to get the final positional embedding.

**Decision block.** Based the comprehensive expression $s^i$ of communication block, decision block will need output the action policy for agent $A^i$ at $t$ time:

$$a_t^i = \text{DecisionBlock}(s_t^i). \tag{5}$$

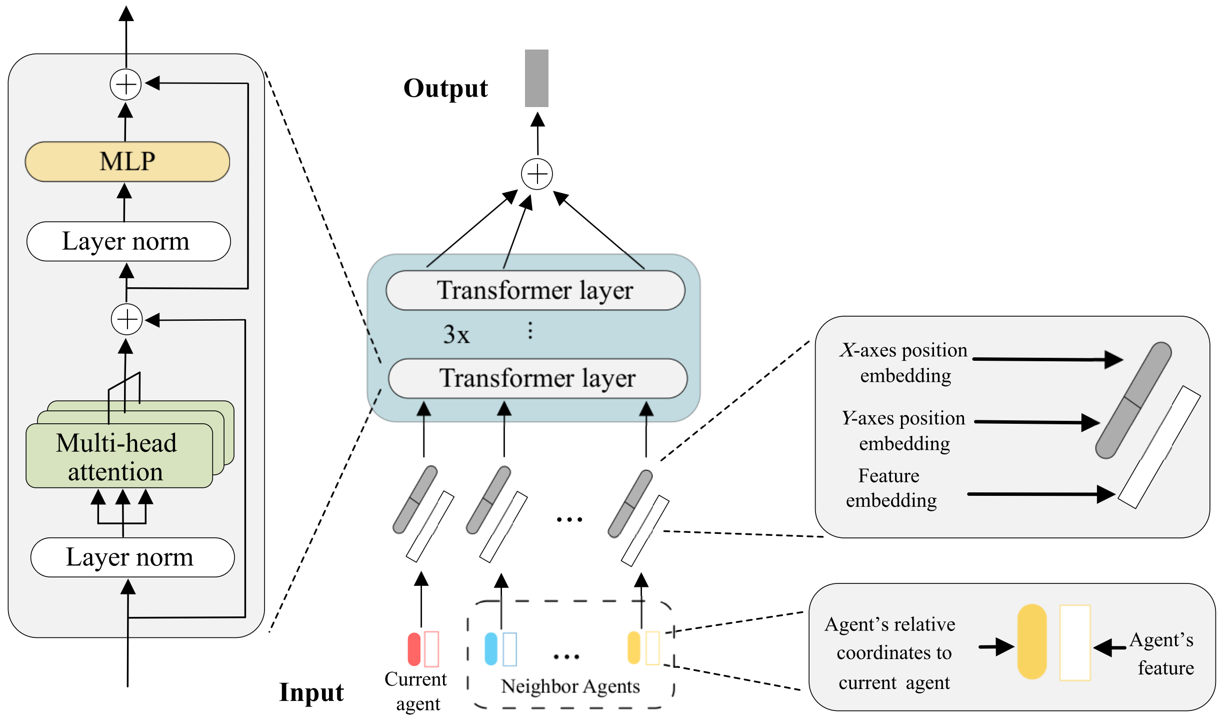


**Fig. 4.** The structure of Transformer-based communication block (TC-Block). The TC-Block takes the features obtained from the LA-Encoder as input, and regards them as a sequence. In addition, there also need to embed the position of each feature tensor. We use relative coordinates as position information. And inspired by Ref. [37] using Transformer in image classification task, we embed the position information from $X$ coordinate and $Y$ coordinate.

We choose SAC[38] to train our model because of its stability and exploratory, and moreover, we compare with other algorithms (e.g., VPG, TRPO, PPO, DDPG, TD3) but they do not achieve the same effect as SAC. It is very hard to prove and explain which one of the popular DRL algorithms, such as VPG, TRPO, PPO, DDPG, TD3, and SAC, is best. And our focus of work does not tend to fully compare the algorithms, so we did not insist that SAC was the best one, while the SAC algorithm may be more suitable for our environment setting. SAC has a parameterized state value function $V_\psi(s_t)$, soft Q function $Q_\theta(s_t,a_t)$, and the strategy function $\pi_\phi$, where $\{\psi,\theta,\ \phi\}$ are their parameters. And SAC is still based on actor-critic architecture, while the value function and the Q function are related, which can be regarded as critic network. The soft value function is trained to minimize the squared residual error:

$$J_V(\psi)=E_{s_t\sim D}\left[\frac{1}{2}(V_\psi(s_t)-E_{a_t\sim\pi_\phi}\ [Q_\theta(s_t,a_t)-\log\pi_\phi(a_t|s_t)])^2\right], \tag{6}$$

where $a_t$ is obtained by current strategy based on the current state $s_t$, $D$ is experience replay buffer, $\log\pi_\phi(a_t|s_t)$ is the entropy of strategy. The soft Q function parameters can be trained to minimize the MSE loss between $Q$ value and target $\hat{Q}$ value:

$$J_Q(\theta)=E_{(s_t,a_t)\sim D}\left[\frac{1}{2}(Q_\theta(s_t,a_t)-\hat{Q}(s_t,a_t))^2\right], \tag{7}$$

where $(s_t,a_t)$ is sampled from experience replay buffer $D$, target $\hat{Q}$ value, and state value $V$ is related:

$$\hat{Q}(s_t,a_t)=R(a_t,s_t)+\gamma E_{s_{t+1}}\ [V_{\bar{\psi}}(s_t)], \tag{8}$$

$V_{\bar{\psi}}$ is the target network which can reduce sample correlation, and the parameter $\bar{\psi}$ of the target value network is updated in a smooth way:

$$\bar{\psi}\leftarrow\tau\psi+(1-\tau)\bar{\psi}, \tag{9}$$

where $\tau$ is the smoothing coefficient.

Finally, the policy parameters $\theta$ can be learned by directly minimizing the expected KL-divergence

$$J_\pi(\phi)=E_{s_t\sim D}\left[D_{KL}(\pi_\phi(\cdot|s_t)||\frac{\exp(Q_\theta(s_t,\cdot))}{Z_\theta(s_t)})\right]. \tag{10}$$

In SAC[38], the updating of actor network $\pi_\phi$ is realized by minimizing KL divergence. SAC uses the reparameterization trick, $a'_t$ is not taken from the experience replay, but the prediction made by reusing the strategy network. The final loss function can be obtained as follows:

$$J_\pi(\phi)=E_{s_t\sim D,\varepsilon_t\sim\Delta}\ [\log\pi_\phi(a'_t|s_t)-Q_\theta(s_t,a'_t)], \tag{11}$$

where $\varepsilon_t$ is noise sampled from Gaussian distribution[38]. The overall procedure is shown in Algorithm 1.

## 4 Results and discussion

### 4.1 Experiment setting

We evaluate our approach in a grid world simulation environment, which is similar to Refs. [15, 16]. The size of the square environment is randomly selected at the beginning of each episode to be either 20, 50, or 100. The obstacle density is randomly selected from 0%, 10%, or 30%. The placement of obstacles, agents, and goals is uniform and random throughout the environment, with the caveat that each agent had to be able to reach its goal. Here, we draw lessons from Ref. [15], and each agent is initially placed in the same connected region as its goal. The actions of the agents are executed sequentially in random order at each time step to ensure that they have equal priority.

Inspired by curriculum learning which is always used in imitation learning, our training procedure is divided into a few stages and starts from easier tasks to more difficult tasks. In the easy scene, 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 number of agents and dynamic obstacles and sample goals in the whole map.

**Algorithm 1:** Off-policy training of proposed framework.

```
1 Initialize the capacity of replay memory $D$;
2 Initialize the weights of observation encoder and decision block;
3 for each iteration do
4     for each environment step $t$ do
5         for each agent $i$ do
6             get local observation $\boldsymbol{L}_t^i$, guiding vector $\boldsymbol{v}_t^i$, and its nearby
                  agents $N_i$;
7             $h_t^i$ = LA-Encoder($\boldsymbol{L}_t^i,\boldsymbol{v}_t^i$);
8             $s_t^i$ = TC-Block($h_t^i,h_t^j$), $j\in N_i$;
9             $a_t^i$ = DecisionBlock($s_t^i$);
10            $s_{t+1}\sim p(s_{t+1}|s_t,a_t)$;
11            $D=D\cup(s_t,a_t,r(s_t,a_t),s_{t+1})$;
12        end for
13    end for
14    for each gradient step do
15        Sample a minibatch data from $D$;
16        Update framework by loss Eq. (6), Eq. (8), and Eq. (11);
17        Update target value network by the smooth Eq. (9);
18    end for
19 end for
```

### 4.2 Training details

We use a discount factor ($\gamma$) of 0.95. We use different length episodes in different size world, e.g. , in 20-size world episode length is 128, in 50-size world episode is 256, in 100-size world episode length is 512. And the batch size is 128 so that integer multiple times of gradient updating can be performed each episode per agent. And we use Pytorch to realize the model and use RAdam[6] with a learning rate beginning at $3\times10^{-4}$.

### 4.3 Metrics

Success Rate = $n_{\text{success}}/n$, where $n_{\text{success}}$ is the number of agents that completed its travel and reached the goal and $n$ is the total number of the agents that need to be planned, is the ratio of the number of agents reaching their goals within a certain time limit over the total number of agents.

Extra Time Rate $= (T - T^*)/T^*$ is the difference between the averaged travel time on all agents and the lower bound of the travel time, where $T$ is the averaged travel time on all agents and $T^*$ is the lower bound of the travel time. The lower bound is the needed time for the agent's tour ignoring other agents.

## 4.4 Baselines

We introduce two traditional algorithms and three learning-based models:

LRA*[10]: Local repair A* (LRA*) is a simple replan method, in which each agent will search for its own path to its goal ignoring other agents, but when an agent will encounter collision the algorithm will recalculate the remainder of its route. LRA* may be an adequate solution for simple environments with few obstacles and few agents. But with more complex environments, LRA* will fall into the dilemma of too many recalculations.

CBS[11]: Conflict-based search (CBS) is a two-level algorithm. At the high level, a search is performed on the conflict tree (CT), which is a tree-based on conflicts between individual agents, whose each node represents a set of constraints for the agents' motion. And at the low level, fast single agent searches are performed to meet the constraints imposed by the high level CT node.

PRIMAL[15, 16]: PRIMAL is a hybrid learning-based method for MAPF that uses both imitation learning (based on an expert algorithm) and multi-agent reinforcement learning. But in PRIMAL it does not take inter-agent communication into consideration. As mentioned in Ref. [18], the key to solving the MAPF problem is learning what, how, and when to communicate.

IL_GNN[18, 19]: They use a convolutional neural network to extract features from local observations, and a graph neural network to communicate these features among agents. But in their work, they use imitation learning to train models, so there will be a lack of exploration of the environment and it's hard to converge.

G2RL[36]: This algorithm combines the global guidance path and trains the model through the framework of reinforcement learning. In G2RL, it does not consider inter-agent communication and add the trick of guide path into every agent's state.

## 4.5 Results

In this section, we will discuss the experimental results of our model and other models in terms of success rate and ETR. At the same time, to better compare the results, we also visualized the results.

**Success rate.** The widely used evaluation index in MAPF is the Success rate, that is, the proportion of the number of agents that have reached the target in all agents within the given time steps of the experiment. The results of the success rate can be found in Table 2. For better observation and comparison, we also visualize results in Fig. 5.

In our experiment, we take a search-based method (LRA*) and a conflict-based method (CBS) as traditional models to be part of baseline. Compared to traditional models, the success rate of our model is very similar to that of the two baselines when the grid world is small. With the growth of the number of agents, world scale, and obstacle density, we can find that the success rate of baseline has decreased obviously, but the success rate of our model decreases more slowly than the two baselines. However, we find that the traditional methods can get the best success rate in small scale world, which is caused by the fact that if the world is small and the number of agent is small, it may be no need to do any communication in this situation. And what's more, if the world is small and the targets are very close to the agents, the information of FOV already contains most of the information of the environment, so the decentralized partial observation problem can be regarded as centralized full observation problem. Then our

**Table 2.** Results for success rate. We compare our LACRL with LRA*[10], CBS[11], PRIAML[15, 16], IL GNN[18], and G2RL[36] on different environment settings. Values are listed as "mean" across 100 instances. And the highest (best) values are highlighted.

| Environment setting | | | Success rate | | | | | |
|---|---|---|---|---|---|---|---|---|
| Map size | Agents' number | Obstacle density | LRA* | CBS | PRIMAL | IL_GNN | G2RL | LACRL |
| 20×20 | 4 | 0% | **1.00** | **1.00** | 0.965 | 0.960 | 0.990 | 0.990 |
| | | 10% | **0.985** | **0.985** | 0.875 | 0.915 | 0.960 | **0.985** |
| | | 30% | 0.975 | **0.985** | 0.920 | 0.910 | 0.955 | **0.985** |
| 50×50 | 25 | 0% | 0.986 | **0.992** | 0.746 | 0.848 | 0.852 | 0.988 |
| | | 10% | 0.972 | **0.990** | 0.774 | 0.926 | 0.826 | **0.990** |
| | | 30% | 0.986 | 0.984 | 0.816 | 0.924 | 0.830 | **0.988** |
| 100×100 | 100 | 0% | 0.961 | 0.974 | 0.678 | 0.896 | 0.783 | **0.981** |
| | | 10% | 0.962 | 0.973 | 0.681 | 0.894 | 0.783 | **0.975** |
| | | 30% | 0.950 | 0.966 | 0.662 | 0.868 | 0.764 | **0.970** |
| 200×200 | 400 | 0% | 0.815 | 0.879 | 0.451 | 0.736 | 0.728 | **0.921** |
| | | 10% | 0.794 | 0.795 | 0.427 | 0.763 | 0.743 | **0.911** |
| | | 30% | 0.772 | 0.754 | 0.464 | 0.776 | 0.719 | **0.890** |

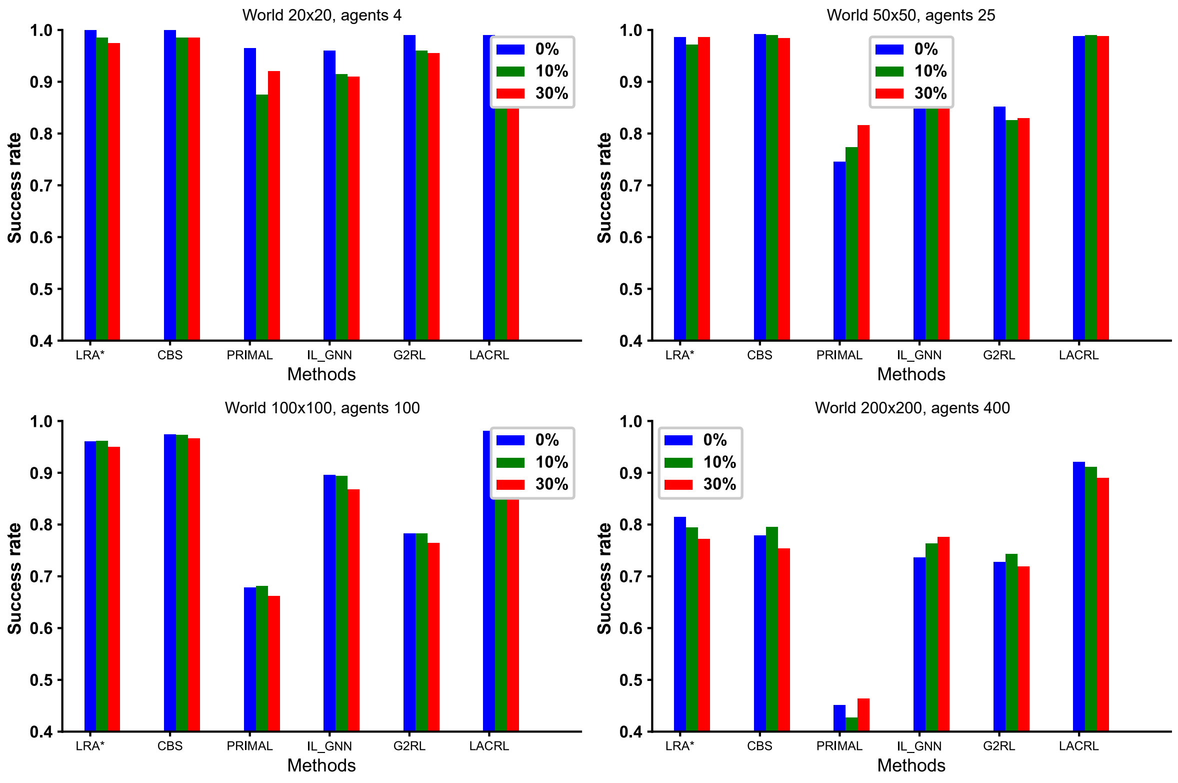


**Fig. 5.** Visual results of success rate. On each subgraph, we list the success rate results of our LACRL and other methods on different obstacle densities. And the higher the column in the histogram, the better the performance of the corresponding model.

model may lose its advantages and will not achieve the high success rate of the traditional methods. Fortunately, the application scenarios in reality will not be particularly small. Besides search-based and conflict-based methods, we also take some learning-based model into consideration (PRIMAL, IL_GNN, G2RL). Compared with the previous learning based models, our model has the best success rate in any case. There is also an interesting discovery that as the problem becomes more complex, the models without the global guide path (PRIMAL) will fast decline, while the models with the global guide path (IL_GNN, G2RL) will slowly decline.

**Extra time rate (ETR).** From the calculation formula of the success rate, it is not difficult to see that the success rate only focuses on whether the task is completed within the specified time, and ignores the quality of the path found. Therefore, we measure the quality of the model strategy according to the proportion of extra time required, that is, the ratio of extra time. Therefore, we use the extra time rate to indicate how much extra time the method needs, in which the low bound of a path length is referred to a global path computed in a static environment ignoring other agents. If the path length found through the model is closer to the global path, we think the solution will be better and the extra time rate metric will be smaller.

The results of extra time rate can be found in Table 3, and the visualization of results are shown in Fig. 6 for better observation and comparison. Obviously, we found that our model LACRL can reach the lowest extra time rate than other learning-based methods in Table 3. And in Fig. 6, we can see that the red line (our LACRL) is always lower than other lines. When the environment scale is small and the number of agents is small, the gap between the red line and other broken lines is small. The first reason for this phenomenon is the environment. Due to the limitation of the environment, even if the solution obtained by the model is very poor, there will be no very high additional time ratio; Secondly, due to the small environment and the small number of agents, it is not difficult for each model to explore the environment, so the difference of additional time ratio will not be too high. This shows that when the scale of the environment becomes larger and the number of agents increases, the advantages and disadvantages of the ability of the model to explore the environment are gradually reflected. It can be roughly seen that the thickness of the shadow with LACRL is always the thinnest, that is, the standard deviation of the model is the smallest, which also shows that our LACRL has a certain stability. In other words, in the same environment, the strategy obtained by the model will not fluctuate greatly in the additional time ratio, which makes the model strategy more stable.

## 4.6 Ablation study

In addition to comparing with the benchmark models, we also conduct ablation experiments that remove or replace part of components of our method. In ablation study, we evaluate the

**Table 3.** Results for extra time rate. Values are listed as “mean / standard deviation” across 100 instances. The lowest (best) values are highlighted. The lower the mean value, the better the model effect. And the smaller the standard deviation, the more stable the model effect.

| Environment setting | | | Extra time rate | | | |
|---|---|---|---|---|---|---|
| Map size | Agents number | Obstacle density | PRIMAL | IL_GNN | G2RL | LACRL |
| 20×20 | 4 | 0% | 0.275 / 0.041 | 0.317 / 0.048 | 0.262 / 0.062 | **0.198 / 0.014** |
| | | 10% | 0.417 / 0.058 | 0.308 / 0.039 | 0.367 / 0.038 | **0.250 / 0.021** |
| | | 20% | 0.428 / 0.062 | 0.447 / 0.037 | 0.356 / 0.042 | **0.329 / 0.019** |
| | | 30% | 0.525 / 0.072 | 0.438 / 0.072 | 0.385 / 0.083 | **0.316 / 0.028** |
| | | 40% | 0.563 / 0.070 | 0.485 / 0.050 | 0.518 / 0.049 | **0.378 / 0.033** |
| 50×50 | 25 | 0% | 0.487 / 0.071 | 0.421 / 0.079 | 0.508 / 0.091 | **0.309 / 0.051** |
| | | 10% | 0.621 / 0.089 | 0.482 / 0.075 | 0.497 / 0.089 | **0.375 / 0.064** |
| | | 20% | 0.596 / 0.086 | 0.462 / 0.084 | 0.542 / 0.086 | **0.424 / 0.071** |
| | | 30% | 0.647 / 0.095 | 0.488 / 0.091 | 0.602 / 0.102 | **0.455 / 0.070** |
| | | 40% | 0.724 / 0.120 | 0.631 / 0.095 | 0.613 / 0.115 | **0.501 / 0.065** |
| 100×100 | 100 | 0% | 0.915 / 0.161 | 0.968 / 0.219 | 0.806 / 0.231 | **0.613 / 0.106** |
| | | 10% | 0.981 / 0.219 | 0.959 / 0.223 | 0.865 / 0.219 | **0.681 / 0.119** |
| | | 20% | 1.124 / 0.171 | 0.984 / 0.171 | 0.943 / 0.171 | **0.707 / 0.121** |
| | | 30% | 1.187 / 0.167 | 1.074 / 0.192 | 1.175 / 0.192 | **0.698 / 0.144** |
| | | 40% | 1.386 / 0.243 | 1.278 / 0.220 | 1.224 / 0.220 | **0.874 / 0.156** |
| 200×200 | 400 | 0% | 1.425 / 0.412 | 1.337 / 0.469 | 1.512 / 0.431 | **0.858 / 0.202** |
| | | 10% | 1.797 / 0.419 | 1.538 / 0.415 | 1.427 / 0.540 | **0.957 / 0.241** |
| | | 20% | 1.548 / 0.421 | 1.767 / 0.494 | 1.575 / 0.402 | **0.910 / 0.255** |
| | | 30% | 2.175 / 0.450 | 1.792 / 0.531 | 1.769 / 0.542 | **1.103 / 0.281** |
| | | 40% | 2.493 / 0.443 | 2.195 / 0.575 | 2.358 / 0.473 | **1.388 / 0.316** |

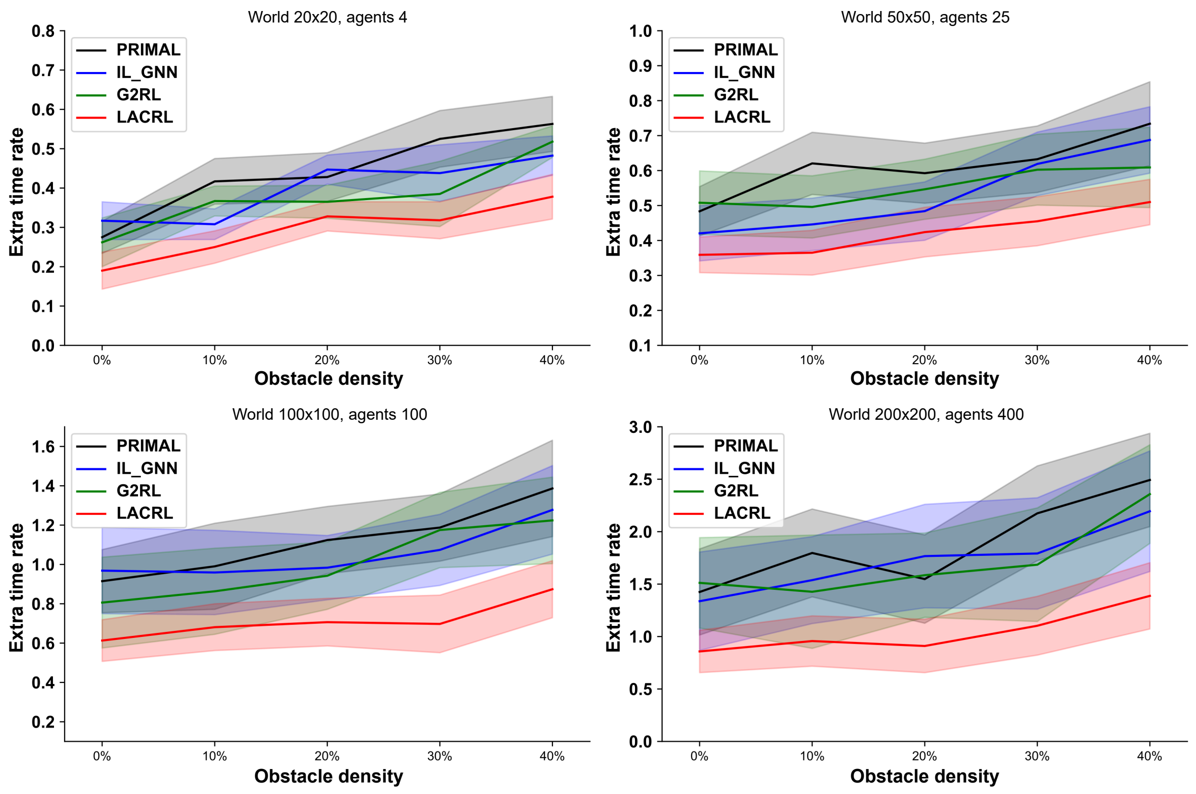


**Fig. 6.** Visual results of extra time rate. On each subgraph, we list the extra time rate results of our LACRL and other methods on different environment settings. And in each line graph, the lower the line, the better the performance of the corresponding model.

performance of our LACRL, LACRL with CNN instead of local attention layer in LA-Encoder (LACRL w/o att.), and LACRL without the communication block (LACRL w/o comm.).

As we can see in Table 4, we find that the results of LACRL with CNN instead of local attention layer (LACRL w/o att.) and LACRL without the communication block (LACRL w/o comm.) have a certain attenuation relative to LACRL. Especially, the attenuation amplitude of LACRL without the communication block is the most obvious. And we find that when the environment size is small, the influence of communication block is small, and the influence of communication block increases with the increase of environment size. Because when the world is small scale and simple, there is no need to do long-horizon decision and the agent will quickly achieve its goal after a little exploration. Therefore, the need of communication is small in that case. However, with the increase of the scale of the environment and the number of agents, the impact of communication block on the experimental performance is becoming greater and greater.Inaword,itisconvincingthatthecommunicationblockand local attention works in MAPF task from ablation study.

## 5 Conclusions

This paper proposes a new model for multi-agent path finding in the partially observable environment, which is formalized into DEC-POMDP process and trained in the form of deep reinforcement learning through repeated trial and error. We build the local observation encoder by using residual attention CNN to extract local observations, and use the transformer architecture to build an interaction layer to combine local observations of agents. With the purpose of overcoming the deficiency of success rate, we also designed a new evaluation index, namely extra time rate (ETR). The experiment results show that our model outperforms traditional methods in most cases, and outperforms the previous learning-based models in all cases in terms of the success rate and the extra time rate among various experiment settings. What's more, the ablation experiment shows that the components exactly work and the performance of our model will deteriorate without them.

Although the performance of our model in the simulated environment is acceptable, there is still a lot of work to be done to expand the experiment to the real environment. The discretization and modeling of the real environment, reward and punishment design and other environmental setting problems need to be adjusted appropriately according to the real environment and practical problems. Therefore, it is still very promising to apply the model in this paper to the actual scene. Later, the experiment can be extended to the actual scene to solve the actual needs.

**Table 4.** Results of ablation study. We evaluate the performance of our LACRL, LACRL without local attention layer in LA-Encoder (LACRL w/o att.), and LACRL without the communication block (LACRL w/o comm.) on different environment settings.

| Environment setting | | | Success rate | | |
|---|---|---|---|---|---|
| Map size | Agents number | Obstacle density | LACRL | LACRL w/o att. | LACRL w/o comm. |
| | | 0% | 0.990 | 0.970 | 0.945 |
| 20×20 | 4 | 10% | 0.985 | 0.960 | 0.920 |
| | | 30% | 0.985 | 0.965 | 0.915 |
| | | 0% | 0.988 | 0.930 | 0.892 |
| 50×50 | 25 | 10% | 0.990 | 0.914 | 0.838 |
| | | 30% | 0.988 | 0.902 | 0.852 |
| | | 0% | 0.981 | 0.880 | 0.771 |
| 100×100 | 100 | 10% | 0.975 | 0.869 | 0.714 |
| | | 30% | 0.970 | 0.853 | 0.726 |
| | | 0% | 0.921 | 0.820 | 0.627 |
| 200×200 | 400 | 10% | 0.911 | 0.863 | 0.661 |
| | | 30% | 0.890 | 0.844 | 0.632 |

## Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities (WK2150110017).

## Conflict of interest

The authors declare that they have no conflict of interest.

## Biographies

**Jinchao Ma** received the B.S. degree in Mathematics from Shandong University in 2019. He was pursuing the M.E. degree at the School of Computer Science and Technology, University of Science and Technology of China. His current research interests include reinforcement learning and path finding algorithm.

**Defu Lian** received his Ph.D. degree in Computer Science from the University of Science and Technology of China (USTC) in 2014. He is currently a Professor in the School of Data Science, USTC. His current research interests include spatial data mining, recommender systems, and learning to hash.

## References

[1] Stern R, Sturtevant N, Felner A, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. arXiv: 1906.08291, **2019**.

[2] Hönig W, Kiesel S, Tinka A, et al. Persistent and robust execution of MAPF schedules in warehouses. *IEEE Robotics and Automation Letters*, **2019**, *4* (2): 1125–1131.

[3] Wurman P R, D'Andrea R, Mountz M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses.. *AI Magazine*, **2008**, *29*: 9–19.

[4] Balakrishnan H, Jung Y. A framework for coordinated surface operations planning at Dallas-Fort Worth international airport. In: AIAA Guidance, Navigation and Control Conference and Exhibit. Hilton Head, USA: AIAA, **2007**: 6553.

[5] Baxter J L, Burke E, Garibaldi J, et al. Multi-robot search and rescue: A potential field based approach. *Studies in Computational Intelligence*, **2007**, *76*: 9–16.

[6] Zhang Y, Qian Y, Yao Y, et al. Learning to cooperate: Application of deep reinforcement learning for online AGV path finding. In: AAMAS '20: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. Auckland, New Zealand: ACM, **2020**: 2077–2079.

[7] Corrales P A, Gregori F A. Swarm AGV optimization using deep reinforcement learning. In: MLMI '20: Proceedings of the 2020 3rd

International Conference on Machine Learning and Machine Intelligence. New York: ACM, **2020**: 65–69.

[8] Yu J, LaValle S. Structure and intractability of optimal multi-robot path planning on graphs. In: AAAI'13: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence. New York: ACM, **2013**: 1443–1449.

[9] Panait L, Luke S. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems,* **2005**, *11*: 387–434.

[10] Silver D. Cooperative pathfinding. In: First Artificial Intelligence and Interactive Digital Entertainment Conference. Washington, DC, USA: AAAI, **2005**: 117–122.

[11] Sharon G, Stern R, Felner A, et al. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence,* **2015**, *219*: 40–66.

[12] Sharon G, Stern R, Felner A, et al. Conflict-based search for optimal multi-agent path finding. In: Proceedings of the International Symposium on Combinatorial Search. Palo Alto, USA: AAAI, **2012**: 190.

[13] Barer M, Sharon G, Stern R, et al. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: Proceedings of the Seventh Annual Symposium on Combinatorial Search. Prague, Czech: AAAI, **2021**, *5*: 19–27.

[14] Boyarski E, Felner A, Stern R, et al. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In: IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence. New York: ACM, **2015**: 740–746.

[15] Sartoretti G, Kerr J, Shi Y, et al. PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters,* **2019**, *4*: 2378–2385.

[16] Damani M, Luo Z, Wenzel E, et al. PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters,* **2021**, *6*: 2666–2673.

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

[18] Li Q, Gama F, Ribeiro A, et al. Graph neural networks for decentralized multi-robot path planning. arXiv: 1912.06095, **2019**.

[19] Li Q, Lin W, Liu Z, et al. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters,* **2021**, *6* (3): 5533–5540.

[20] Liu Z, Chen B, Zhou H, et al. MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, USA: IEEE, **2020**: 11748–11754.

[21] Jansen R, Sturtevant N. A new approach to cooperative pathfinding. In: AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems. New York: ACM. **2008**: 1401–1404.

[22] Ryan M R K. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research,* **2008**, *31*: 497–542.

[23] Wang K H C, Botea A. Fast and memory-efficient multi-agent pathfinding. In: ICAPS'08: Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling. New York: ACM, **2008**: 380–387.

[24] Aljalaud F, Sturtevant N R. Finding bounded suboptimal multi-agent path planning solutions using increasing cost tree search. In: Proceedings of the Sixth International Symposium on Combinatorial Search. Washington, DC, USA: AAAI, **2013:** 203.

[25] Sharon G, Stern R, Goldenberg M, et al. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence,* **2013**, *195*: 470–495.

[26] Walker T T, Sturtevant N R, Felner A. Extended increasing cost tree search for non-unit cost domains. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence Main track. California: IJCAI, **2018**: 534–540.

[27] Surynek P. On propositional encodings of cooperative path-finding. In: 2012 IEEE 24th International Conference on Tools with Artificial Intelligence. Athens, Greece: IEEE, **2013**: 524–531.

[28] Surynek P. Compact representations of cooperative path-finding as SAT based on matchings in bipartite graphs. In: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence. Limassol, Cyprus: IEEE, **2014**: 875–882.

[29] Surynek P. Reduced time-expansion graphs and goal decomposition for solving cooperative path finding sub-optimally. In: IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence. New York: ACM, **2015**: 1916–1922.

[30] Surynek P, Felner A, Stern R, et al. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In: ECAI'16: Proceedings of the Twenty-Second European Conference on Artificial Intelligence. New York: ACM, **2016**: 810–818.

[31] Yu J, LaValle S M. Multi-agent path planning and network flow. In: Frazzoli E, Lozano-Perez T, Roy N, et al., editors. Algorithmic Foundations of Robotics X. Berlin: Springer, **2013**: 157–173.

[32] Yu J, LaValle S M. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics,* **2016**, *32* (5): 1163–1177.

[33] Lam E, Le Bodic P, Harabor D, et al. Branch-and-cut-and-price for multi-agent pathfinding. *Computers & Operations Research,* **2022**, *144*: 105809.

[34] Erdem E, Kisa D, Oztok U, et al. A general formal framework for pathfinding problems with multiple agents. *Proceedings of the AAAI Conference on Artificial Intelligence,* **2013**, *27* (1): 290–296.

[35] Chen Y F, Liu M, Everett M, et al. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore: IEEE, **2017**: 285–292.

[36] Wang B, Liu Z, Li Q, et al. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters,* **2020**, *5* (4): 6932–6939.

[37] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16×16 words: Transformers for image recognition at scale. arXiv: 2010.11929, **2020**.

[38] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv: 1801.01290, **2018**.