Heuristic Large-Scale Multi-Agent Path Planning with Communication

Wei Pang[1*] , Degang Wang[1]
1. College of Control Science and Engineering, Dalian University of Technology, Dalian, China
pangw2022@163.com, wangdg@dlut.edu.cn
Corresponding Author: Wei Pang Email: pangw2022@163.com

2024 International Conference on New Trends in Computational Intelligence (NTCI) | 979-8-3315-1702-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/NTCI64025.2024.10776364

***Abstract*—Multi-agent path planning is an important topic in robotics and has significant implications in autonomous vehicles and intelligent logistics. In this paper, a heuristic multi-agent path planning method is proposed. The improved channel and spatial attention modules are utilized to extract key features from the maps of agents. Then communication is conducted through the spatio-temporal Graph Transformer, enhancing coordination among agents. To improve training efficiency, the heuristic path generated by expert algorithm is designed for model training. In addition, a distance-based function is applied to assist agents in escaping deadlock situations. Experimental results demonstrate that the proposed method performs well in multi-agent path planning tasks across various maps.**

Keywords—multi-agent path planning; heuristic; communication; attention

I. INTRODUCTION

In recent years, advances in robotics have provided a solid foundation for the construction of large-scale multi-agent systems. Multi-agent path planning is important in areas such as formation control, logistics, and warehouse automation. The state space grows exponentially as the number of agents increases, which makes path planning extremely complex. Additionally, multi-agent path planning requires the interactions and coordination among agents to avoid collisions and conflicts. Therefore, the construction of efficient and fast multi-agent path planning methods has become a hot topic in this field.

Various heuristic strategies and communication mechanisms have been proposed to optimize the path planning process. Some methods focus on improving searching efficiency and path quality by introducing heuristic guidance. The shortest path from a single source is used as a heuristic guide in [1], which effectively guides the agents to avoid invalid paths. The impact of the next action for agents is further considered in [2], where four possible actions are embedded into a heuristic map, leading to a significant improvement in efficiency. A novel heuristic-based attention mechanism is designed in [3] to pick out critical information and enhance collaboration among agents. Additionally, in [4], a conflict-based search algorithm is used as a heuristic guidance to generate paths and enable distributed path planning.

In addition to heuristic strategies, how to establish an effective communication mechanism is also a crucial approach for solving path planning problems. In [5], graph convolutional network is utilized to create communication network among agents. A method combining graph neural networks and the key-query mechanism is proposed in [6] to improve the effectiveness of communication between agents. To improve communication efficiency and reduce computational burden, a request-response communication mechanisms in [7] is designed which can allow agents to actively choose to communicate with important neighbors.

Although existing methods have been widely applied in multi-agent path planning, how to design an effective path planning model is still an interesting question. To handle this problem, a heuristic multi-agent path planning model with communication is proposed. The model utilizes enhanced channel and spatial attention module to improve the recognition of map features. Spatial and temporal information are further dynamically integrated through a spatio-temporal Graph Transformer, resulting in more efficient communication. The rest contents are organized as follows. A comprehensive overview of the multi-agent path planning network with communication is given in Section II. Section III presents the multi-agent path planning algorithm based on heuristic path. Section IV evaluates the performance of the proposed algorithm through various numerical experiments. The conclusion is summarized in Section V.

II. COMMUNICATION-BASED MULTI-AGENT PATH PLANNING NETWORK

In this section, the communication-based multi-agent path planning network is designed to improve coordination among agents. The basic structure of the network is shown in Fig. 1. The network input is the maps of agents. Channel and spatial attention are used to extract map features, and communication is facilitated by the spatio-temporal Graph Transformer to achieve path planning.

The map contains extensive environmental information, such as the locations of obstacles, neighboring agents, and target positions. The channel and spatial attention modules ([8]) are utilized to extract map features. By adaptively assigning different weights to each channel and dynamically adjusting their importance, the channel attention is considered to avoid the interference from

irrelevant data by focusing on critical information on the map.

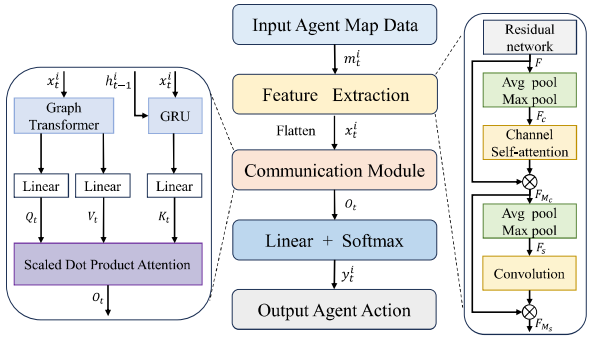

Fig. 1. Architecture of multi-agent path planning network

At time t, the map data $\boldsymbol{M} = [m_t^1, m_t^2, \cdots, m_t^N]$ is represented as the input to the network, and $m_t^i \in \mathbb{R}^{3\times W_{FOV}\times H_{FOV}}$ is a binary tensor with values 0 and 1, where 0 represents empty spaces, 1 represents obstacles or agents. The input data is preprocessed by a three-layer residual network to obtain the feature $\boldsymbol{F} \in \mathbb{R}^{C\times W\times H}$.

In the max pooling and average pooling layers of channel attention, the pooled feature $\boldsymbol{F}_c \in \mathbb{R}^{C\times 1\times 1}$ is generated from feature $\boldsymbol{F}$:

$$\boldsymbol{F}_c = \frac{1}{H\cdot W}\sum_{i=1}^{H}\sum_{j=1}^{W}\boldsymbol{F}(i,j) + \max_{i\in[1,H],j\in[1,W]}\boldsymbol{F}(i,j) \quad (1)$$

where H and W are the dimensions of the input feature $\boldsymbol{F}$, i and j represent elements in feature $\boldsymbol{F}$.

Then pooled feature $\boldsymbol{F}_c$ undergoes a linear transformation in the channel self-attention layer, facilitating the dynamic adjustment of the weights of different channels:

$$\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{F}_c, \boldsymbol{K} = \boldsymbol{W}_k\boldsymbol{F}_c, \boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{F}_c \quad (2)$$

where $\boldsymbol{W_q} \in \mathbb{R}^{C\times C}$, $\boldsymbol{W}_k \in \mathbb{R}^{C\times C}$ and $\boldsymbol{W}_v \in \mathbb{R}^{C\times C}$ are the weight parameters. $\boldsymbol{Q} \in \mathbb{R}^{C\times 1\times 1}$ and $\boldsymbol{K} \in \mathbb{R}^{C\times 1\times 1}$ are used to calculate the weights between channels. $\boldsymbol{V} \in \mathbb{R}^{C\times 1\times 1}$ is utilized to combine with the calculated weights. The weights of different channels are calculated as follows:

$$\boldsymbol{M}_c = \phi_{\text{softmax}}\left(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d_k}\right)\boldsymbol{V} \quad (3)$$

where d_k is a scaling factor, $\phi_{\text{softmax}}(\cdot)$ denotes the softmax function, and $\boldsymbol{M}_c \in \mathbb{R}^{C\times 1\times 1}$ is the attention weight map. Then the channel attention output $\boldsymbol{F}_{\boldsymbol{M}_c} \in \mathbb{R}^{C\times W\times H}$ is obtained as follows:

$$\boldsymbol{F}_{\boldsymbol{M}_c} = \boldsymbol{M}_c \odot \boldsymbol{F} \quad (4)$$

where $\odot$ is Hadamard product.

Subsequently, the output $\boldsymbol{F}_{\boldsymbol{M}_c}$ is taken as the input for spatial attention. Pooling layers of spatial attention compress the channel dimension of $\boldsymbol{F}_{\boldsymbol{M}_c}$:

$$\boldsymbol{F}_s = \left[\frac{1}{C}\sum_{c=1}^{C}\boldsymbol{F}_{\boldsymbol{M}_c}(c), \max_{c\in[1,C]}\boldsymbol{F}_{\boldsymbol{M}_c}(c)\right] \quad (5)$$

where c is the channel dimension of the input $\boldsymbol{F}_{\boldsymbol{M}_c}$. Then the compressed features $\boldsymbol{F}_s \in \mathbb{R}^{2\times W\times H}$ are integrated in convolution layer:

$$\boldsymbol{M}_s = \phi_{\text{sigmoid}}(\boldsymbol{W}_{conv} * \boldsymbol{F}_s) \quad (6)$$

where $\phi_{\text{sigmoid}}(\cdot)$ denotes the sigmoid function and $*$ reprsents the convolution operation, $\boldsymbol{W}_{conv} \in \mathbb{R}^{1\times 2\times k}$ is the weight parameters of the convolutional layer, k represents the convolution kernel size, $\boldsymbol{M}_s \in \mathbb{R}^{1\times W\times H}$ is the spatial attention map. The output of spatial attention $\boldsymbol{F}_{\boldsymbol{M}_s} \in \mathbb{R}^{C\times W\times H}$ is calculated in equation (7), then the output $\boldsymbol{F}_{\boldsymbol{M}_s}$ is flattened into the feature vector $x_t^i \in \mathbb{R}^{1\times F}$.

$$\boldsymbol{F}_{\boldsymbol{M}_s} = \boldsymbol{M}_s \odot \boldsymbol{F}_{\boldsymbol{M}_c} \quad (7)$$

After applying the channel and spatial attention modules, the extracted map feature is passed as input of the communication module. Subsequently, the construction process of the communication module is presented.

The Graph Transformer ([9]) is used to construct the communication network, which effectively capturing spatial information. In addition, a Gated Recurrent Unit (GRU) ([10]) is incorporated to capture temporal information.

The input of the Graph Transformer is the extracted map features $\{x_t^1, x_t^2, \cdots, x_t^N\}$, where N denotes the number of agents. The weight coefficient $\alpha_{m,ij,t}$ is calculated as follows:

$$q_{m,t}^i = \boldsymbol{W}_{m,q,t}x_t^i + b_{m,q,t}, k_{m,t}^j = \boldsymbol{W}_{m,k,t}x_t^j + b_{m,k,t} \quad (8)$$

$$\alpha_{m,ij,t} = \frac{\langle q_{m,t}^i, k_{m,t}^j\rangle}{\sum_{u\in N(i)}\langle q_{m,t}^i, k_{m,t}^u\rangle} \quad (9)$$

where $q_{m,t}^i \in \mathbb{R}^{1\times F}$ and $k_{m,t}^j \in \mathbb{R}^{1\times F}$ are vectors used to calculate the feature similarity between agent i and agent j, $\boldsymbol{W}_{m,q,t} \in \mathbb{R}^{F\times F}$ and $\boldsymbol{W}_{m,k,t} \in \mathbb{R}^{F\times F}$ are weight parameters, $b_{m,q,t} \in \mathbb{R}^{1\times F}$ and $b_{m,k,t} \in \mathbb{R}^{1\times F}$ are bias parameters. $\langle q_{m,t}^i, k_{m,t}^j\rangle = \exp(q_{m,t}^{i\,T}k_{m,t}^j/\sqrt{d})$ and d is the hidden size. $N(i)$ is the set of neighboring agents of agent i. The message aggregation from agent j to agent i is as follows:

$$v_{m,t}^j = \boldsymbol{W}_{m,v,t}x_t^j + b_{m,v,t} \quad (10)$$

$$\hat{x}_t^i = \frac{1}{M}\sum_{m=1}^{M}\sum_{j\in\mathcal{N}(i)}\alpha_{m,ij,t}v_{m,t}^j \quad (11)$$

where $\hat{x}_t^i \in \mathbb{R}^{1\times F}$ is the final spatial feature, the feature x_t^j is transformed to vector $v_{m,t}^j$ for weighted sum, $\boldsymbol{W}_{m,v,t} \in \mathbb{R}^{F\times F}$ and $b_{m,v,t} \in \mathbb{R}^{1\times F}$ are trainable parameters used for transformation, M is the number of Graph Transformer attention heads, and $\alpha_{m,ij,t}$ is the weight coefficient.

The input values of the GRU are the map feature $x_t^i \in \mathbb{R}^{1\times F}$ and the previous hidden state $h_{t-1}^i \in \mathbb{R}^{1\times F}$. After the

reset gate and update gate, the temporal information $h_t^i \in \mathbb{R}^F$ is obtained as follows:

$$h_t^i = z_t^i \odot h_{t-1}^i + (1 - z_t^i) \odot \tilde{h}_t^i \tag{12}$$

$$z_t^i = \sigma(\boldsymbol{W}_{xz} x_t^i + \boldsymbol{W}_{hz} h_{t-1}^i + b_z) \tag{13}$$

$$\tilde{h}_t^i = \tanh(\boldsymbol{W}_{xh} x_t^i + (r_t^i \odot h_{t-1}^i)\boldsymbol{W}_{hh} + b_h) \tag{14}$$

$$r_t^i = \sigma(\boldsymbol{W}_{xr} x_t^i + \boldsymbol{W}_{hr} h_{t-1}^i + b_r) \tag{15}$$

where $\boldsymbol{W}_{xr} \in \mathbb{R}^{F\times F}$, $\boldsymbol{W}_{hr} \in \mathbb{R}^{F\times F}$ and $b_r \in \mathbb{R}^F$ are paremeters of reset gate. $\boldsymbol{W}_{xz} \in \mathbb{R}^{F\times F}$, $\boldsymbol{W}_{hz} \in \mathbb{R}^{F\times F}$ and $b_z \in \mathbb{R}^F$ are parameters of update gate, $\odot$ is Hadamard product, and $\sigma(\cdot)$ and $\tanh(\cdot)$ are respectively sigmoid and hyperbolic tangent functions.

The temporal information $\boldsymbol{H}_t = [h_t^1, h_t^2, \cdots, h_t^N]$ and spatial information $\widehat{\boldsymbol{X}}_t = [\hat{x}_t^1, \hat{x}_t^2, \cdots, \hat{x}_t^N]$ obtained from the Graph Transformer and GRU are dynamically fused, the calculations for the linear layers in the dynamic fusion are as follows:

$$\begin{aligned} \boldsymbol{Q}_t &= \boldsymbol{H}_t \boldsymbol{W}'_q + \boldsymbol{b}_q, \\ \boldsymbol{K}_t &= \widehat{\boldsymbol{X}}_t \boldsymbol{W}'_k + \boldsymbol{b}_k \\ \boldsymbol{V}_t &= \widehat{\boldsymbol{X}}_t \boldsymbol{W}'_v + \boldsymbol{b}_v \end{aligned} \tag{16}$$

where $\boldsymbol{W}'_q, \boldsymbol{W}'_k, \boldsymbol{W}'_v \in \mathbb{R}^{F\times D}$ are weight parameters and $\boldsymbol{b}_q, \boldsymbol{b}_k, \boldsymbol{b}_v \in \mathbb{R}^{N\times D}$ are bias parameters of the linear layers. The temporal information $\boldsymbol{H}_t$ is converted into the query matrix $\boldsymbol{Q}_t \in \mathbb{R}^{N\times D}$. The spatial information $\widehat{\boldsymbol{X}}_t$ is converted into the key matrix $\boldsymbol{K}_t \in \mathbb{R}^{N\times D}$ and value matrix $\boldsymbol{V}_t \in \mathbb{R}^{N\times D}$. The fused output $\boldsymbol{O}_t \in \mathbb{R}^{N\times D}$ is obtained in scaled dot product attention layer:

$$\boldsymbol{O}_t = \phi_{softmax}(\boldsymbol{Q}_t {\boldsymbol{K}_t}^{\mathrm{T}} / \sqrt{D}) \boldsymbol{V}_t \tag{17}$$

where D is the hidden size of the linear layers. In the scaled dot product attention layer, the similarity between the temporal information and spatial information is computed using the query matrix $\boldsymbol{Q}_t$ and the key matrix $\boldsymbol{K}_t$ to obtain dynamic weights. Accordingly, the weights are applied to scale the value matrix $\boldsymbol{V}_t$, resulting in the fused output $\boldsymbol{O}_t$.

III. Heuristic Multi-Agent Path Planning Algorithm

This section presents the multi-agent heuristic path planning algorithm. Each agent has a start position and a target position on maps. The expert algorithm (EECBS) ([11]) is used to generate the next action v_t^i at each time step t, resulting in a series of conflict-free expert-level paths:

$$V^* = \{V^1, V^2, \cdots, V^N\} \tag{18}$$

where N is the number of agents, $V^i = \{v_{t_1}^i, v_{t_2}^i, \cdots, v_{t_n}^i\}$ is the heuristic path for agent i, V^* is the set of paths for agents.

Similarly, the corresponding map information at each time step is retained as:

$$M^* = \{M^1, M^2, \cdots, M^N\} \tag{19}$$

where $M^i = \{m_{t_1}^i, m_{t_2}^i, \cdots, m_{t_n}^i\}$ is a series of surrounding maps for agent i, M^* is the set of maps for all agents. Then, the heuristic paths V^* of all agents and the corresponding maps M^* are collected in the expert dataset.

The map information is used as input to the multi-agent path planning model, and the predicted action y_t^i is provided by a decision network based on a multi-layer perceptron. The multi-agent path planning network is trained by minimizing the loss between the predicted actions y_t^i and the expert actions v_t^i, gradually optimizing the path planning capabilities of agents. The cross-entropy loss is expressed as follows:

$$\mathcal{L}(v_t^i, y_t^i) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{n} v_t^i \log(y_t^i) \tag{20}$$

where y_t^i is the action predicted by model, and v_t^i is the action directed by expert. The loss function accurately quantifies the deviation between the predicted path and the optimal path which can provide an intuitive accuracy indicator.

The optimal network parameters are obtained by minimizing the cross-entropy function. The AdamW optimization algorithm ([12]) is used to solve this problem.

Due to path conflicts, unclear priorities, or unreachable paths, some agents may become deadlocked during path planning. Deadlocks can lead to planning failures. To address this issue, it is necessary to detect whether an agent is in a deadlock state. An agent is considered to be in a deadlock if either of the following situations occurs during path planning:

1) The agent remains at the non-target position for over three time steps;
2) The agent hovers between two positions for more than four time steps.

A method using a distance-based function to unlock deadlock is proposed. It indicates the next action for the agent by calculating the distance between four positions around it and the target position. The distance-based function is defined as follows:

$$h(m, n) = |x_n - x_m| + |y_n - y_m| \tag{21}$$

where (x_n, y_n) and (x_m, y_m) are the coordinates of the agent positions on maps.

In Fig. 2, agent i encounters two types of deadlock situations. For the first type of deadlock, the distance-based function calculates the distance value from the surrounding positions to the target. The position with the shortest distance is selected as the next action, guiding the agent out of the deadlock.

For the second type of deadlock, there are two positions with the shortest distance: up and right. If agent i moves up, it will continue to be stuck in the deadlock, so this position needs to be excluded. When the second type of deadlock

occurs, the agent is prohibited from repeating the action from the previous time step. Instead, the agent should choose the other direction with the shortest distance.

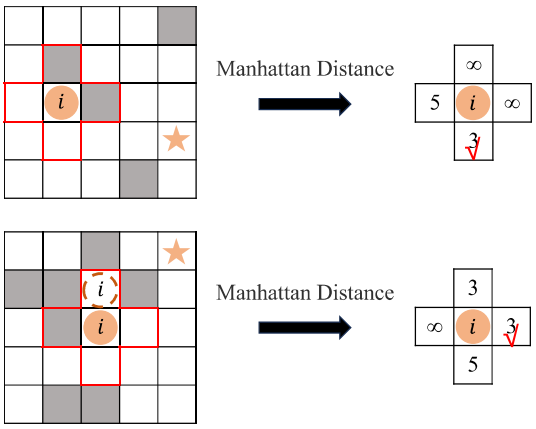

Fig. 2. Distance-based function to unlock deadlocks

The steps of the proposed algorithm are summarized as follows:

- Step 1. Generate the heuristic path v_t^i and the map information m_t^i, and randomly initialize the model parameters $\boldsymbol{W}$.
- Step 2. Input the map data into the channel and spatial attention modules to generate the extracted map feature x_t^i based on equations (1) to (7).
- Step 3. Input the extracted map feature into the Graph Transformer and GRU. Generate spatial information $\hat{x}_t^i$ and temporal information h_t^i according to equations (8) to (15). Dynamically integrating temporal and spatial information according to equations (16) and (17).
- Step 4. Pass the fused features $\boldsymbol{O}_t$ through a fully connected layer to produce the predicted action y_t^i.
- Step 5. Compute the cross-entropy loss $\mathcal{L}$ according to equation (20). Optimize the parameters using the AdamW optimization algorithm.
- Step 6. Calculate the distance according to equation (21) when an agent is in deadlock state. Choose the shortest distance as the next action.
- Step 7. The model training will stop when the predefined number of iterations is reached, otherwise it will continue.

IV. Experiment

The experimental platform is Ubuntu 20.04 LTS, with an Intel i5-13600KF CPU, an NVIDIA GeForce RTX 4060 Ti GPU, 16 GB of GPU memory, and 32 GB of system memory. The development environment for the experiment is PyCharm 2023.2.1, with Python 3.9.18 and PyTorch 2.0.1 as the programming language and deep learning framework, respectively.

The dataset of 2D grid maps with varying sizes and agent numbers is used in the experiments. For the first series of experiments, obstacles on the maps are randomly distributed and the obstacle density is 10%. Start and goal positions are randomly assigned. In the dataset, the map sizes include 20×20, 28×28, 35×35, 40×40, 45×45, 50×50 and 65×65, while the number of agents varies among 10, 20, 30, 40, 50, 60 and 100. Sixty percent of the samples are randomly selected for training, twenty percent for validation, and the remaining samples are used for testing. Table 1 presents the simulation parameters used in the experiment.

TABLE 1. Algorithm Simulation Parameter Configuration Table

Para-meters	Values	Description
F	128	The dimensions of the extracted features
k	7	The size of the convolutional kernel
M	4	Number of Graph Transformer attention heads
D	128	The hidden size the linear layers
N_D	200	The total training epochs
E	5	Interval of validation epochs
α	0.001	The learning rate of the optimizer
λ	1e-5	Optimizer weight decay coefficient

In the first series of experiments, the proposed method is compared with three other models: GNN ([4]), MAGAT ([6]), and HiMAP ([13]). Learning rate of the GNN model is chosen to be 10^{-3}, with batch size of 64 and communication feature dimension is 128. The MAGAT model consists of four heads communication layer, learning rate is chosen to be 10^{-3}, and batch size is 64. The HiMAP model uses a temperature softmax layer, 35 training epochs, and learning rate of 5×10^{-5}.

TABLE 2. Success Rates for Different Maps

Maps (Agents)	Success Rate			
	HiMap	GNN	MAGAT	Ours
20×20 (10)	0.921	0.958	**0.981**	**0.981**
28×28 (20)	0.883	0.941	0.978	**0.980**
35×35 (30)	0.850	0.914	0.975	**0.977**
40×40 (40)	0.745	0.868	0.960	**0.971**
45×45 (50)	0.769	0.844	0.958	**0.963**
65×65 (100)	0.664	0.770	0.941	**0.956**
50×50 (20)	0.846	0.962	0.989	**0.992**
50×50 (30)	0.850	0.941	0.979	**0.984**
50×50 (40)	0.795	0.909	0.972	**0.979**
50×50 (50)	0.803	0.864	0.962	**0.975**
50×50 (60)	0.776	0.861	**0.973**	0.950
50×50 (100)	0.527	0.510	0.884	**0.891**

TABLE 3. MAKE SPAN FOR DIFFERENT MAPS

Maps (Agents)	Make Span			
	HiMap	GNN	MAGAT	Ours
20×20 (10)	85.81	28.92	**28.20**	28.44
28×28 (20)	112.32	47.20	43.22	**43.03**
35×35 (30)	130.05	64.61	56.30	**55.57**
40×40 (40)	141.82	81.67	68.23	**67.65**
45×45 (50)	156.78	97.97	78.45	**76.67**
65×65 (100)	224.29	168.56	123.71	**123.11**
50×50 (20)	133.73	76.24	71.99	**71.24**
50×50 (30)	146.79	83.39	76.79	**76.67**
50×50 (40)	163.44	91.72	80.87	**80.03**
50×50 (50)	164.45	103.20	85.06	**84.24**
50×50 (60)	172.57	109.26	**86.20**	93.54
50×50 (100)	195.02	181.75	115.06	**112.98**

Tables 2 and 3 present the average simulation results for the success rate and make span over 1,000 cases on each map. The results demonstrate that the proposed method is better than the other three methods. These facts mean that multi-agent path planning network which incorporates channel and spatial attention modules can effectively extract critical features from maps and provide agents with precise environmental information. In addition, the use of a distance-based deadlock resolution method assists agents in overcoming obstacles.

TABLE 4. AVERAGE STEPS FOR DIFFERENT MAPS

Agents	Map size 40×40		Map size 80×80	
	DHC [1]	Ours	DHC [1]	Ours
4	**52.33**	53.58	**96.72**	96.80
8	63.90	**61.89**	**109.24**	111.42
16	79.63	**76.86**	122.54	**122.47**
32	100.10	**98.63**	138.32	**136.15**
64	147.26	**144.65**	163.50	**162.04**

In the second series of experiments, the obstacle density is 30%. The map sizes include 40×40 and 80×80. The number of agents varies among 4, 8, 16, 32, 64. Sixty percent of the samples are randomly selected for training, twenty percent for validation, and the remaining samples are used for testing. Training parameters are set as $N_D = 50$ and $E = 2$, with other settings consistent with Table 1.

The proposed method is compared with DHC ([1]). The average experimental results for 200 cases on each map are shown in Table 4. It can be seen that the proposed method requires fewer average steps than the DHC model. The comparative results indicate that the communication module based on the spatio-temporal Graph Transformer enables agents to share information and coordinate their actions. This global information sharing mechanism reduces the number of steps required for successful path planning.

V. CONCLUSIONS

A heuristic multi-agent path planning method is proposed in this paper. The algorithm utilizes the spatio-temporal graph transformer to improve communication efficiency and employs heuristic paths to accelerate model training, resulting in satisfactory performance. In future research, we will focus on multi-agent path planning methods under conditions of communication delay or limited communication among agents.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (2018AAA0100300) and the National Natural Science Foundation (NNSF) of China (12071056).

REFERENCES

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

[2] Y. Wang, B. Xiang, S. Huang and G. Sartoretti, "SCRIMP: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding," in 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 9301-9308, 2023.

[3] Q. Lin and H. Ma, "SACHA: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding," in IEEE Robotics and Automation Letters, vol. 8, no. 8, pp. 5100-5107, Aug. 2023.

[4] Q. Li, F. Gama, A. Ribeiro and A. Prorok, "Graph neural networks for decentralized multi-robot path planning, " in 2020 IEEE/RSJ international conference on intelligent robots and systems, 2020, pp. 11785-11792.

[5] Z. Ye, Y. Li, R. Guo, J. Gao, and W. Fu, "Multi-agent pathfinding with communication reinforcement learning and deadlock detection," in International Conference on Intelligent Robotics and Applications, pp. 493-504, 2022.

[6] Q. Li, W. Lin, Z. Liu and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," in IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5533-5540, July 2021.

[7] Z. Ma, Y. Luo and J. Pan, "Learning selective communication for multi-agent path finding," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 1455-1462, April 2022.

[8] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in Proceedings of the European conference on computer vision, pp. 3-19, 2018.

[9] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," arxiv preprint arxiv:2009.03509, 2020.

[10] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arxiv preprint arxiv:1406.1078, 2014.

[11] J. Li, W. Ruml and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in Proceedings of the AAAI conference on artificial intelligence, pp. 12353-12362, 2021.

[12] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arxiv preprint arxiv:1711.05101, 2017.

[13] H. Tang, F. Berto, Z. Ma, C. Hua, K. Ahn and J. Park, "HiMAP: Learning heuristics-informed policies for large-scale multi-agent pathfinding," in Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, pp. 2498-2500, 2024.