Proceedings of the ASME 2023
International Mechanical Engineering Congress and Exposition
IMECE2023
October 29-November 2, 2023, New Orleans, Louisiana

IMECE2023-113113

A DECENTRALIZED MULTI-AGENT PATH PLANNING APPROACH BASED ON IMITATION LEARNING AND GLOBAL STATIC FEATURE EXTRACTION

Bohan Feng[1] **Youyi Bi**[1*] **Mian Li**[2] **Liyong Lin**[3]

[1] University of Michigan – Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University
Shanghai, China

[2] Global Institute of Future Technology
Shanghai Jiao Tong University
Shanghai, China

[3] Contemporary Amperex Technology Co., Limited,
No. 2, Xingang Road,
Zhangwan Town,
Jiaocheng District, Ningde,
Fujian, China

ABSTRACT

Multi-agent path planning is a crucial problem in numerous industrial robotic implementations, ranging from smart material transportation, mobile patrolling to automated warehousing. In this paper, we introduce a decentralized multi-agent path planning approach based on imitation learning and global static feature extraction. Our approach employs a convolutional neural network in the information extraction layer to obtain features from local field-of-view observations and expert-planned paths under a global static map. The information aggregation layer then uses a graph attention network to combine feature information from selected neighboring agents. The request-reply based selective communication is also applied in the information aggregation layer to identify appropriate neighboring agents to be included in the graph attention network. Finally, the action output layer translates the aggregated feature information into actions for each agent. Additionally, we develop a strategy switching mechanism that adaptively utilizes expert-planned paths under a global static map to support agents to escape from local traps. The effectiveness of our proposed approach is evaluated in simulated grid environments with varying map sizes, obstacle densities, and numbers of agents. Experimental results demonstrate that our approach outperforms other decentralized path planning methods in success rate and generalizability. Furthermore, our approach is computationally efficient and scalable, making it suitable for real-world applications.

Keywords: Decentralized control, Mobile robot, Multi-agent Path Planning, Selective communication

1. INTRODUCTION

In the era of Industry 4.0, mobile robots are playing a significant role in transforming various aspects of modern manufacturing, such as mobile assembly, smart material transportation, and production line patrolling [1,2]. Consequently, the development of efficient multi-agent path planning (MAPP) algorithms is crucial to ensure these robots to work safely and properly [3].

The aim of MAPP is to generate collision-free paths for multiple agents to move from their initial positions to target positions on a given map. Traditional MAPP typically adopts a centralized approach, in which a central planning unit generates paths for all agents based on global information. Centralized methods can offer optimality and completeness in their solutions, ensuring that the best possible path is found for each agent if it exists [4,5]. However, these methods suffer from scalability issues as the number of robots increases, making them less suitable for applications with large-scale robots in complex environments [3].

Recent studies explore to develop decentralized MAPP methods in which each agent computes its own path and make movement decisions based on local information [6,7]. In the case of conflicts, only the affected agents need to replan their paths. Decentralized methods significantly reduce the computational burden of the central planning unit, making it more scalable for large-scale systems. Nevertheless, decentralized methods may not always produce optimal solutions since they prioritize generating efficient paths for individual agents with a lack of

* Corresponding author, Assistant Professor in Mechanical Engineering, Shanghai Jiao Tong University
Email: youyi.bi@sjtu.edu.cn

considering the overall system efficiency. These methods also face the problem of limited successful deployment in complex scenarios as they cannot fully exploit the information from global static environment, e.g., the result of single-agent path planning in the global static map. This information can assist with more comprehensive perception of local environment features and support more efficient communication among neighboring agents.

Therefore, in this paper we propose a novel decentralized multi-agent path planning approach based on imitation learning and global static feature extraction. The core idea is to utilize a learning-based architecture integrated with global static feature information to enhance the accuracy and practicality of decentralized path planning without compromising its scalability in complex environments. The effectiveness of the proposed approach is validated in simulated grid environments with varying map sizes, obstacle densities, and numbers of agents. The experiment results show that our approach outperforms other decentralized path planning methods in success rate and generalizability. The primary contribution of this paper includes:

- A decentralized multi-agent path planning approach based on imitation learning and global static feature extraction is proposed.
- A graph attention network to combine feature information from request-reply based selective communication procedure is designed.
- A strategy switching mechanism that adaptively utilizes expert-planned paths under a global static map to support agents to escape from local traps is developed.

The rest of the paper is structured as follows: Section 2 presents a literature review of related work in multi-agent path planning. Section 3 introduces the problem formulation. Section 4 shows the overall architecture of our approach and explains the key techniques involved, including the three layers of imitation learning and the strategy switching mechanism. Section 5 compares the performance of our approach with other state-of-the-art algorithms in a simulated grid environment. Lastly, Section 6 provides a summary of our study and highlights potential directions for future research.

2. RELATED WORK

MAPP methods can be classified into centralized and decentralized. In centralized methods, a central planning unit calculates coordinated waypoints for all robots based on their locations and destinations. The robots then use this information for real-time navigation. By considering all robots sharing a common state space, centralized methods can provide optimal and complete path plans, but their computational demand is high when the working environment for robots is complex. For example, conflict-based search (CBS) and its variant Enhanced CBS (ECBS) [4,8] can find optimal or suboptimal path solutions in which the high-level central unit searches for a set of collision constraints and impose these constraints on individual agents. M* and its variant [5] extend the standard A* algorithm to generate paths for each agent and apply a sub-dimensional expansion strategy to dynamically increase the dimensionality of the search space in regions where agent collisions occur. The operator decomposition (OD) and M* are combined by OD-recursive-M* (ODrM*) [9] to keep the branching factor small during the search, which reduces the agents that require joint planning. Despite some progress has been made in reducing the computational load, these centralized methods still struggle to handle environments with numerous potential path conflicts.

In contrast to centralized methods, decentralized MAPP methods rely less on a central unit and more on individual agents to make independent planning decisions based on local environment information. Optimal reciprocal collision avoidance (ORCA) [10] is a classical decentralized method, which assumes that each agent has complete knowledge of the shape, position, and velocity of its neighbor agents. Using this information, the robot computes its velocity to ensure safe planning within its next time step. In recent years, learning-based decentralized MAPP algorithms are receiving higher interest. Researchers explored to use imitation learning or reinforcement learning to train robots to move independently according to locally sensed information [11]. PRIMAL [6] is a classical decentralized MAPP framework based on reinforcement learning and imitation learning, allowing robots to plan paths and coordinate in real-time within a partially observable world. Li et al. [7] demonstrate the potential of graph neural networks (GNNs) in learning explicit communication policies for complex multi-robot coordination. They also use imitation learning to enable decentralized algorithms to approximate the performance of centralized expert algorithms. Distributed, Heuristic and Communication (DHC) [12] combines graph neural networks with deep Q-learning to improve policy performance in complex obstacle environments. Chen et al. [13] introduce a new framework based on imitation learning and reinforcement learning that combines transformer structures, contrastive learning, and dual deep Q networks to achieve feasible MAPP in dense environments without requiring inter-robot communication.

In the field of large-scale robotic systems, the ability of learning-based approaches to handle high-dimensional state-space representations is of particular interest. One advantage of learning-based decentralized planning is that each robot can gather information from other nearby robots, enabling more efficient coordination. However, current learning-based algorithms have not adequately addressed the following important questions:

- What feature information should be effectively shared among agents? While effective communication is key to decentralized path planning, it is not yet obvious what information is critical to the planning and what information should be shared among agents. Due to the complexity of MAPP problems, the optimal strategy of coordination is unknown and the rule-based coordination usually cannot provide the required performance [11]. Existing learning-based decentralized MAPP algorithms mainly rely on the guidance of target directions for path planning, and they often ignore the auxiliary information from global static

map in local feature extraction. By making full use of this auxiliary information, it is expected that the success rate of learning-based decentralized MAPP algorithms can be further improved.

- Is the feature information of all neighboring agents equally important? Most existing learning-based MAPP algorithms employ broadcasting communication to all neighboring agents. In fact, frequent communications lead to high communication and computational burden. Researchers have proposed several methods to alleviate this issue, such as Attention Mechanism [14], Temporal Message Control (TMC) [15], Individually Inferred Communication (I2C) [16], Decision Causal Communication [17], etc. While some of these methods can adjust information passing among agents, they may struggle to effectively deal with redundant or repetitive data that does not add new insights into the decision-making process of path planning. It is still not fully clear how the communication between neighbors can be further optimized according to the local feature information.
- How can local traps be effectively handled in learning-based path planning? Although existing learning-based algorithms have achieved good performance in many scenarios, it is still challenging for them to achieve a 100% success rate [13]. A typical kind of planning failure is local trap, in which a robot oscillates around a few positions as its outward passages are blocked by obstacles or other robots. One possible solution to this issue is to utilize expert-planned paths under a global static map to provide guidance for these agents to escape from local traps.

In this paper, we expect to preliminarily address the above issues by introducing a decentralized MAPP approach based on imitation learning and global static feature extraction. The global static feature information is incorporated into the local feature extraction for agents, and guides the selection of communication between agents within a certain range. With the support of a strategy switching mechanism using local expert-planned path information based on global static feature, the success rate of learning-based MAPP can be further improved.

3. PROBLEM FORMULATION

In our MAPP research, the focus is placed on 2D grid maps (see Fig.1), characterized by 4-neighbor connectivity. In this setting, each entity (i.e., an agent or an obstacle) occupies a single grid cell. The map is represented by an $l_w \times l_h$ graph matrix, with 0 representing a free vertex and 1 indicating an occupation vertex. For each map, we choose the starting and corresponding goal vertices for M agents from the available free positions. Each goal vertex is reachable from its starting vertex, and there is no overlap among the $2M$ chosen positions. Time is divided into discrete intervals, referred to as time steps. At each time step t, agents can move to adjacent positions or wait at their current vertices, resulting in an action space with a size of five (i.e., move upward, downward, left, right or remain stationary). The path for agent i is defined as a series of connected vertices (i.e., moving) or identical vertices (i.e., waiting). After reaching their goal vertices, agents maintain their positions. We consider a partially observable environment, which is more representative in real-world scenarios. Each agent can access information only within its perceptual space. The perceptual space of agent i at time t, denoted by o_i^t, is the sensing area of size $r_w \times r_h$ centered at the agent's position. Here, $r_w < l_w$ and $r_h < l_h$ represent the width and height of the perceptual space are smaller than that of the whole map. During operation, the agent should avoid collisions with obstacles or other agents. The goal of MAPP is to find a set of paths devoid of collisions for each agent. The solution's quality is evaluated based on the maximal arrival time steps of all agents at their respective goal vertices.

4. METHODS

4.1 The Overall Structure of The Proposed Approach

Figure 1 shows the overall workflow of the proposed decentralized MAPP approach. Our approach adopts an imitation learning architecture, consisting of three major layers: information extraction, information aggregation and action output. In imitation learning, by leveraging expert demonstrations, agents can rapidly learn complex behaviors, significantly reducing the learning time compared to trial-and-error methods such as reinforcement learning. Moreover, imitation learning offers transferability, allowing the learned decentralized policy to be easily adapted in different scenarios.

As depicted in Fig.1, the local observation of agent i is first processed through the local observation space processing. The agent i's local observation is updated at each time step. The obtained observation o_i^t is then fed into the information extraction layer to get $\tilde{o}_i^t$. Subsequently, $\tilde{o}_i^t$ is passed into the information aggregation layer, which aggregates information through the request-reply based selective communication procedure and graph attention network to obtain $\tilde{h}_i^{t(k)}$. Finally, $\tilde{h}_i^{t(k)}$ is directed to the action output layer to generate the action output a_i^t. Additionally, in the application inference stage, we design a strategy switching mechanism that adaptively switches between the expert A* algorithm using the global static map and the learning-based algorithm to help agents escape from local traps. In the following subsections, the detailed explanations of the major components of our approach and the strategy switching mechanism are provided.

4.2 Local Observation Space

We consider a partially observable discrete grid world where each agent has a finite field of view defined by an $r_w \times r_h$ matrix, and the global map is represented by an $l_w \times l_h$ matrix with random static obstacles. Beyond this field of view, the agent cannot perceive any information. Partial observation is crucial for real-world deployment of path planning, as it allows the policy to be generalized to environments with arbitrary sizes and reduces the neural network's input dimensionality. We denote the local observation space as o_i^t. As illustrated in Figure 2, the

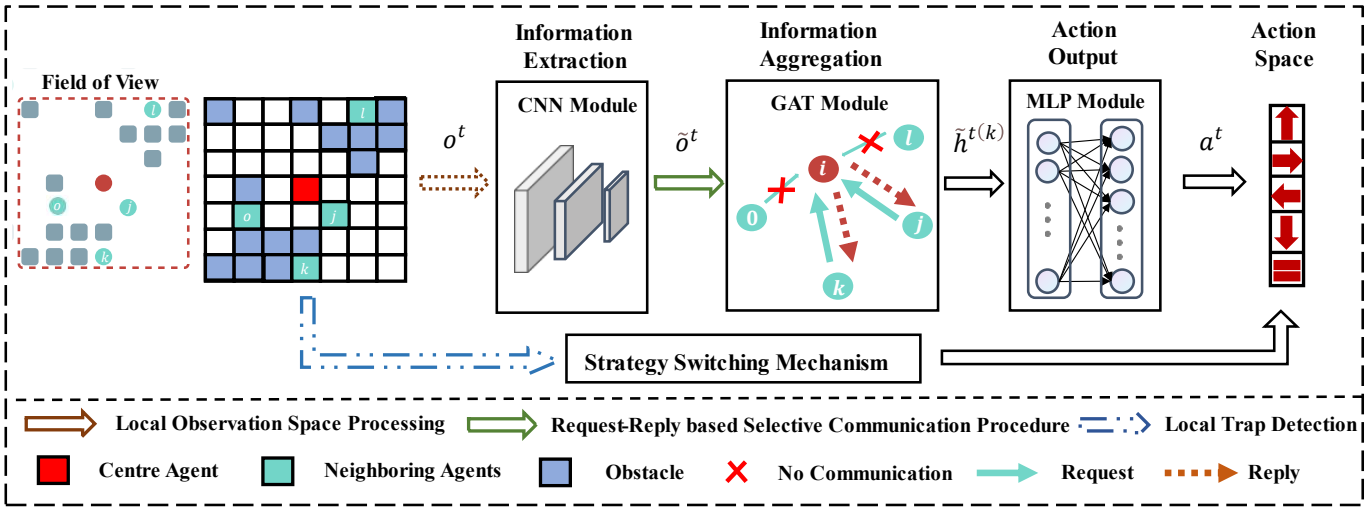

FIGURE 1: The overall workflow of the proposed decentralized MAPP approach.

available information in the finite field of view for path planning is divided into four distinct channels to simplify the agent's learning. Specifically, Channel 1 contains the agent i's local observation of the environment. Channel 2 represents the position of goal $v_{i,g}$, or its projection into the boundary of the local observation $p_{i,g}^t$. Channel 3 displays the intercepted path e_i^t calculated by agent i using the A* algorithm under global static map information (here other agents $j \in M$ are not considered). Channel 4 shows the relative positions of other neighboring agents $j \in N_i$ with respect to agent i. By separating the information into different channels, agents can understand and process their local environment with higher efficiency, leading to improved performance in path planning.

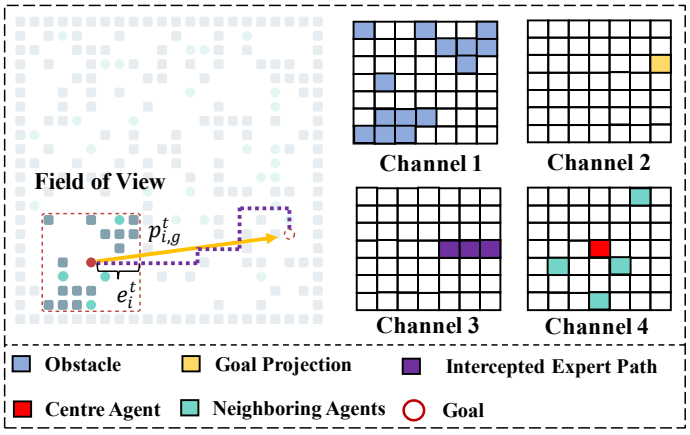

FIGURE 2: The available information in the finite field of view for path planning (i.e., 4 channels). It illustrates how we process the partial observations of each robot.

4.3 Information Extraction Layer

In the information extraction layer, a Convolutional Neural Network (CNN) architecture is leveraged to process the information within local observation space. The key role of CNN is to extract features from the local observation through local receptive fields and weight sharing strategies. This process helps preserve and utilize the spatial structural information inherent in the input observation. Specifically, the input observation feature o_i^t is processed by a CNN operating internally on agent i. Then CNN generates a vector $\tilde{o}_i^t \in \mathbb{R}^H$, which contains H observations ($\tilde{o}_i^t = CNN(o_i^t)$). The observation matrix $\tilde{O}^t$ can be expressed as:

$$\tilde{O}^t = \begin{bmatrix} \tilde{o}_{1,1}^t & \tilde{o}_{1,2}^t & \cdots & \tilde{o}_{1,H}^t \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{o}_{M,1}^t & \tilde{o}_{M,2}^t & \cdots & \tilde{o}_{M,H}^t \end{bmatrix} \in \mathbb{R}^{M \times H} \quad (1)$$

The $\tilde{o}_i^t$ can be subsequently transferred to nearby agents. The primary use of the CNN is to transform the input observation features o_i^t into a higher-level feature $\tilde{o}_i^t$.

Figure 3 shows the CNN architecture for information extraction, which is constructed by three sequentially stacking convolutional modules according to previous research [7]. Each of the three modules comprises Conv2-BatchNorm2d-ReLU-MaxPool layers. This architecture allows efficient processing of local observation information and facilitates effective communication between agents. The generated $\tilde{o}_i^t$ will undergo further processing through information aggregation layer as detailed in the subsequent section.

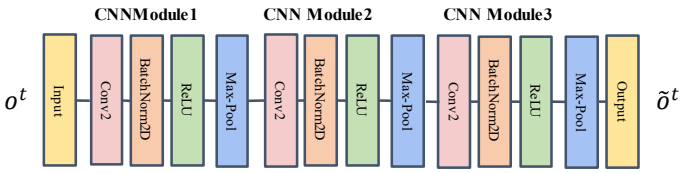

FIGURE 3: The CNN architecture for information extraction.

4.4 Information Aggregation Layer

The information aggregation layer comprises two parts. The first part is a request-reply based selective communication procedure, which aims to reduce the communication burden by narrowing down the communication scope of agents within a certain range. The second part involves the implementation of a graph attention network for feature aggregation with the selected agents.

4.4.1 The Request-Reply Based Selective Communication Procedure

Figure 4 shows the procedure of the request-reply based selective communication. It involves a two-step communication filtering process. The first step is quadrant selection based on the feature information extracted from the global static map. Let $Q_i = \{Q_1, Q_2, Q_3, Q_4\}$ be the four quadrants of the agent i's local field of view. We can then determine a smaller quadrant set Q_i' that contains $p_{i,g}^t$ and e_i^t, in which $p_{i,g}^t$ is the projection of the agent's goal position into the boundary of the agent's local observation and e_i^t is the intercepted path information. Agents situated in Q_i' are assumed to have additional communication value, thereby the set of neighboring agents with necessity of communication for agent i can be narrowed down. Here we use N_i to represent the set of neighboring agents in the local field view of the agent i, and N_i' as the set of agents located in Q_i'. Obviously, the number of neighboring agents that agent i needs to communicate with becomes smaller.

The second step is the request-reply mechanism-based selection. The request-reply mechanism depends on the information extraction layer and the action output layer, to get temporary actions. Agent i gets its local observation o_i^t from the environment and constructs another modified observation $o_{i,-j}^t$ ($-j$ means without agent j, $j \in N_i'$) by setting the information at agent j's position to some special value, such as zero.

Each $o_{i,-j}^t$ is passed through the information extraction layer to get an embedding $\tilde{o}_{i,-j}^t$. By skipping the information aggregation layer, $\tilde{o}_i^t$ and $\tilde{o}_{i,-j}^i$are directly fed into the action output layer to compute action-values. Actions $\tilde{a}_i^t$ and $\tilde{a}_{i,-j}^t$ are inferred by applying argmax function over action-values. If these two actions match with each other, we infer that the existence of agent j will not affect agent i's policy, then agent i will not request communication from agent j. Equation (2) shows the selective communication function for agent i:

$$\rho_{ij} = \begin{cases} 0, & \text{if } \tilde{a}_i^t = \tilde{a}_{i,-j}^t, i \in M, j \in N_i' \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

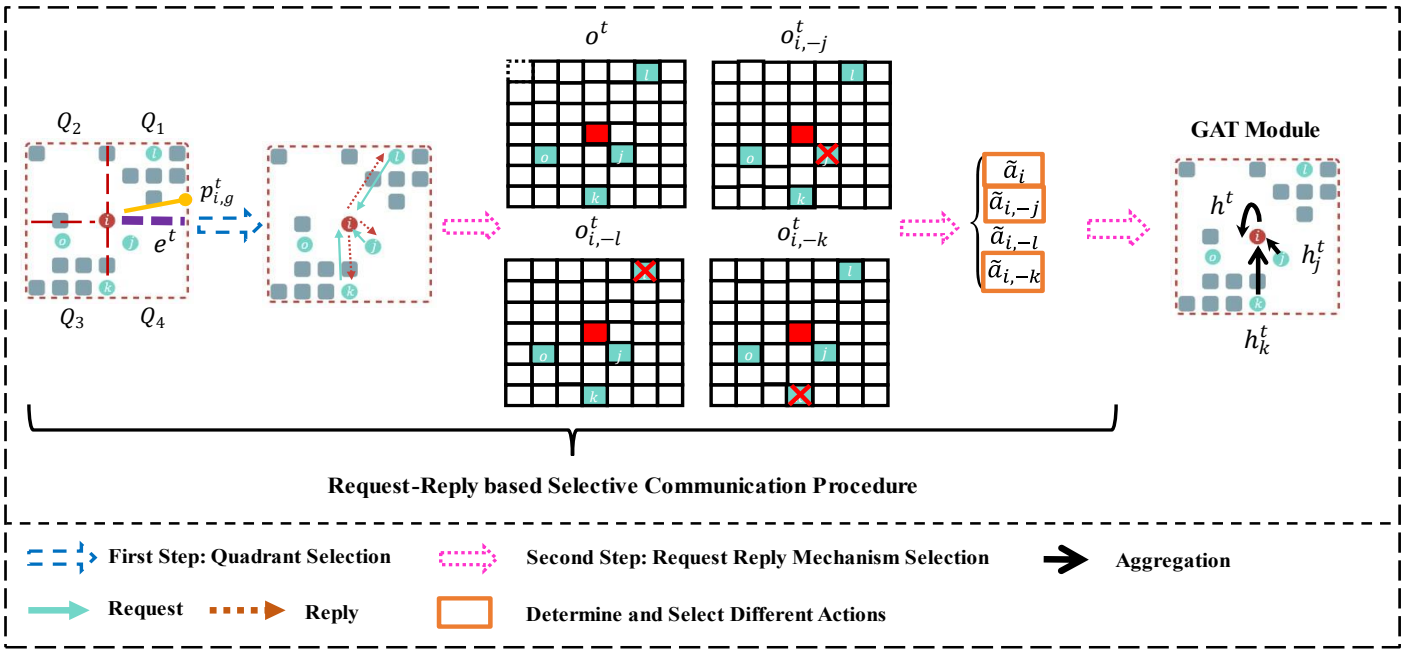

FIGURE 4: An illustration of the request-reply based selective communication procedure. In the first step, the communication range is narrowed down to quadrant $Q_i' = \{Q_1, Q_4\}$ where $p_{i,g}^t$ is located and e_i^t passes. Then the agents located in these quadrants (i.e., agents l, j and k) are assumed to have potential communication value. By inputting o_i^t, $o_{i,-l}^t$, $o_{i,-j}^t$ and $o_{i,-k}^t$ to information extraction layer and action output layer, we can find the difference or similarity between output action $\tilde{a}_i^t$ and $\tilde{a}_{i,-l}^t$, $\tilde{a}_{i,-j}^t$, $\tilde{a}_{i,-k}^t$. This information will be used to determine the necessity of communicating between agent i and other agents (l, j, k), completing the second step of selective communication procedure. Finally, the GAT is used to aggregate the feature information of agents j and k as these two agents have passed the check of the selective communication procedure.

Note that temporary actions $\tilde{a}_i^t$ and $\tilde{a}_{i,-j}^t$ are only used to decide the communication scope, not the final actions to be executed. Then the communication scope of agent i can be determined by the set of agents $\mathcal{S}_i$ as shown in Equation (3):

$$\mathcal{S}_i = \{j \mid \rho_{ij} = 1, i \in M, j \in N_i'\} \tag{3}$$

4.4.2 The Graph Attention Network (GAT)

After getting the neighboring agents with necessity of communication through the selective communication procedure, we use the graph attention network (GAT) [18] to aggregate the feature information of these agents. The goal of the GAT is to update the features of each agent by aggregating information from its neighboring agents, considering both the features of the selected neighbor agents and the edge weights obtained from the attention mechanism.

In the GAT process, we first apply the linear transformation to the feature vector of each agent $\tilde{o}_i^t$ $(i \in M)$, using a shared weight matrix $W \in \mathbb{R}^{H' \times H}$, where H and H' are the input and output feature dimensions, respectively.

$$h_i^t = W(\tilde{o}_i^t)^T \tag{4}$$

The attention mechanism computes the importance of each selected neighboring agent's information to the current agent i. The attention function is denoted by $attention: \mathbb{R}^{H'} \times \mathbb{R}^{H'} \rightarrow \mathbb{R}$, and the attention coefficients e_{ij}^t are computed as:

$$e_{ij}^t = attention(h_i^t, h_j^t) \tag{5}$$

The choice for the attention mechanism $attention$ is a single-layer feedforward neural network with the LeakyReLU activation function:

$$attention(h_i^t, h_j^t) = \text{LeakyReLU}(a^T[h_i^t \oplus h_j^t]) \tag{6}$$

where $a \in \mathbb{R}^{2H'}$ is a trainable weight vector, $.^T$ represents matrix transposition, and $\oplus$ denotes the concatenation of two vectors. The attention coefficients are then normalized using the $softmax$ function:

$$\alpha_{ij}^t = softmax(e_{ij}^t) = \frac{\exp(e_{ij}^t)}{\sum_{k \in \mathcal{S}(i)} \exp(e_{ik}^t)} \tag{7}$$

Now, we can update the features of each agent by aggregating the information from its neighbors, weighted by the normalized attention coefficients:

$$\tilde{h}_i^t = \sigma\left(\sum_{j \in \mathcal{S}(i)} \alpha_{ij}^t h_j^t\right) \tag{8}$$

where $\sigma: \mathrm{R} \rightarrow \mathrm{R}$ denotes a sigmoid activation function. To improve the model's capacity and stability, we use multi-head attention, where we learn K independent attention mechanisms:

$$\tilde{h}_i^{t(k)} = \sigma\left(\sum_{j \in \mathcal{S}(i)} \alpha_{ij}^{t(k)} W^k (\tilde{o}_i^t)^T\right) \tag{9}$$

The final output features can be obtained by concatenating the output of all attention heads:

$$\tilde{h}_i^{t(k)} = \oplus_{k=1}^{K} \tilde{h}_i^{t(k)} \tag{10}$$

where $\oplus$ represents concatenation. Note that, in this setting, the final returned output, $\tilde{h}_i^{t(k)}$, will consist of $K \times H'$ features (rather than H') for each node. The final feature $\tilde{h}_i^{t(k)}$ is further translated into actions through the action output layer as described in the following section.

4.5 Action Output Layer

The final movement decision for agent i at time step t is determined by the action output layer. In this layer, we employ a Multi-Layer Perceptron (MLP) as the decision-making component, i.e., $a_i^t = MLP(\tilde{h}_i^{t(k)})$. The MLP, which shares weights across all agents, receives the output from the information aggregation layer and maps the feature information into action vectors. A $softmax$ function is applied after the MLP layer to convert the action vector into a probability distribution vector for five discrete actions (i.e., move up, down, left, right, and stop). The action with the highest probability is selected as the final output action, in which the argmax function is used to choose the action with the highest probability. The final generated path for the agent i is a set of sequential actions $\{a_i^1, \ldots, a_i^t\}$, which represents the agent i's trajectory from the starting position to the goal position.

4.6 The Training Procedure of Our Approach

In our study, the agents operate in a $l_w \times l_h$ grid world with randomly placed static obstacles. We generate random cases for each grid map, consisting of pairs of starting and goal vertices for all agents. Duplicate or invalid cases are eliminated, and the remaining cases are stored in a pool of training sets, which are randomized during training. For each case, we first use the expert algorithm ECBS to compute the solution. At the beginning of training, we have the expert actions $\{E^*\}$ for all agents and the corresponding local feature observation $\{O^t\}$, collected in the training set $\mathcal{T} = \{(\{E^*\}, \{O^t\})\}$. The predicted action trajectories obtained from the imitation learning network are denoted as $\hat{E}$. We use the cross-entropy loss function $\mathcal{L}(.,.)$ to train the network:

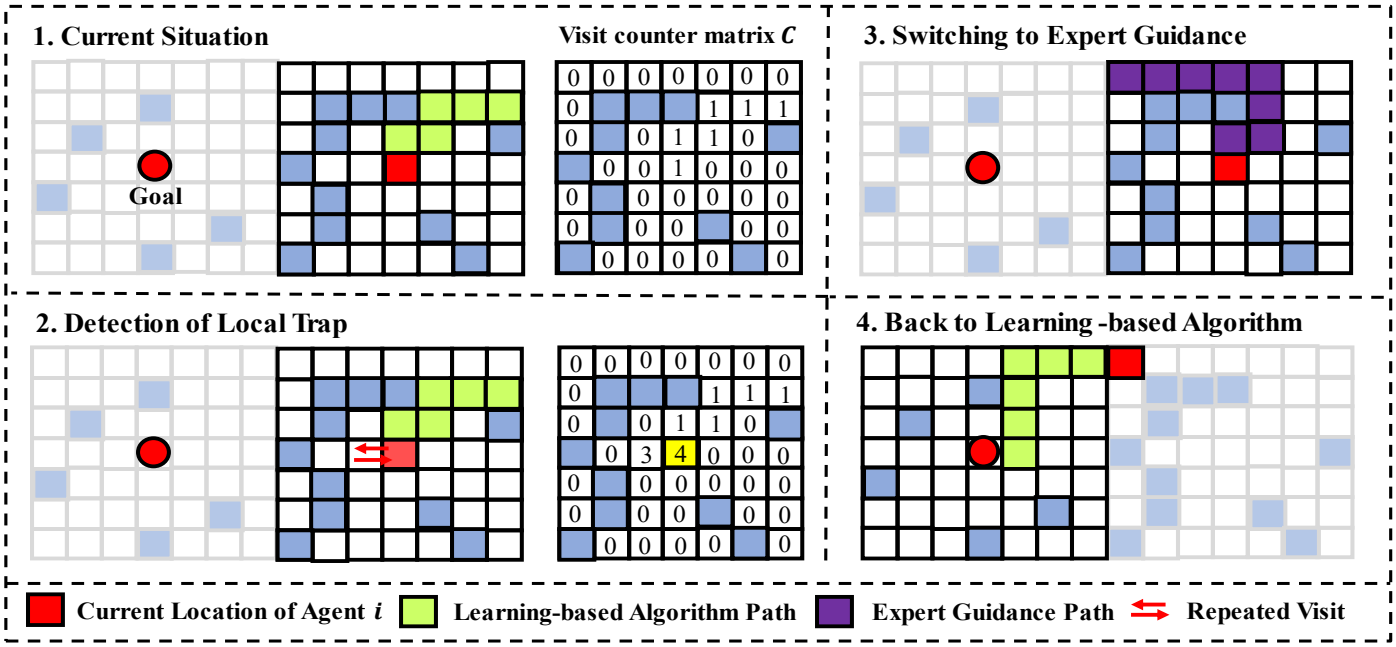

FIGURE 5: An illustration of the strategy switching mechanism. We only show the visit counter matrix C in the local field of view of agent i (see the red dot). As depicted by the yellow marker, if the visit count exceeds the threshold value δ, the system identifies the agent i as being in a local trap. In such case, the agent switches to the expert algorithm for path planning to get out of the trap.

$$\mathcal{L}(\hat{E}, E^*) = -\sum_{t=1}^{T}\sum_{i=1}^{M} E_i^*(o_i^t)\log \hat{E}_i(o_i^t) \qquad (11)$$

where T is the total number of time steps.

During the training process, the collision shielding safety mechanism [7] is utilized to ensure the agent learns collision-free paths. In the event of vertex or edge collisions, the affected agent's action is replaced with an idle action. At that moment, the agent remains idle until it times out, which is considered as a failure case. As deadlocks between agents can occur, using the collision shielding safety mechanism may result in planning failures where some agents remain idle throughout the rest planning process. For every C epochs, we select n random cases from the training set and check these cases. For the failed cases, we use the expert algorithm ECBS to continue planning and obtain successful solutions. The successful cases are then added back to the training set. Additionally, for the successfully planned cases in n random cases, we check whether the planned paths deviate significantly from the expert-planed paths. If the deviation ratio exceeds a threshold value r_d, these cases will be added back to the training set for re-training. This treatment can support the planned paths from our approach to approximate the expert-planned paths as much as possible.

4.7 The Strategy Switching Mechanism

As mentioned earlier, when deploying the trained models in simulated environments, the success rate for agents to reach their goals seldom achieves 100%. This limitation can be attributed to the trained model's struggling with handling complex environments such as local traps. Local traps are areas where agents become stuck or fail to find viable paths due to restricted local conditions, such as dead ends or areas with a high density of obstacles. To mitigate this issue and increase the success rate, we develop a strategy switching mechanism that can adaptively switch between the expert A* algorithm using the global static map and the learning-based algorithm when encountering local traps. As depicted in Figure 5, the mechanism operates as follows:

(1) Detection of local traps: We first design a local trap detector that monitors the agents' states and identifies the trapping situations. This detection is based on the number of repeated visits to a particular grid cell. We create a visit counter matrix C with the same dimensions as the grid-based environment, initialized with zeros. Each element in this matrix corresponds to a grid cell and stores the number of times that the agent has visited to that cell. Then the visit counter matrix will be updated for each time step t. At each step t, the current grid cell's visit count in the matrix will be checked. If the visit count exceeds the threshold δ, it is considered as a local trap.

(2) Switching to expert guidance: Once a local trap is detected, the path planning strategy switches to use expert-planed path guidance (see the purple cells in Fig. 5) generated from A* algorithm for a predetermined number of steps η, until the agent reaches the boundary of its local observation space.

(3) Switching back to learning-based algorithm: If the agent has successfully completed the predetermined number of steps and reaches the boundary of its local observation space, the path planning strategy reverts to the learning-based path planning algorithm. This enables the agent to continue exploring the environment and dynamically adjust its path based on the current state (see the green cells in Fig. 5).

5. EXPERIMENT SETTINGS AND RESULTS

In this section, we first introduce the settings of the model parameters and experiment environment. Then the performance evaluation metrics and different MAPP methods to be compared are introduced. Finally, the experimental results are analyzed and discussed.

5.1 Model Parameters and Environment Settings

In the information extraction layer, a convolutional kernel size of 3 with a stride of 1 and no padding is used. The number of shared features H before the aggregation layer is set to 64. In the information aggregation layer, we use one layer of the graph attention network with a K value of 4 for multi-head attention mechanism. During the network training process, we randomly select $n = 200$ cases from the training set every $C = 5$ epochs to check for failed cases and cases that deviate significantly from expert-planned paths. The Adam optimizer with a momentum of 0.9 is used. A dynamic learning rate is used with a starting value of 10^{-4}, which is decreased by 50% at 200 epochs and 400 epochs, respectively. For the application inference phase, we set the visit count threshold value δ to 4.

According to the environment settings in previous research [7,14], we initialized 500 different maps of size $l_w \times l_h = 20 \times 20$, with 70% of them being used for training, 15% for validation, and 15% for testing. Furthermore, each map contains 40 randomly placed obstacles. It's also worth mentioning that each map generates 60 cases, with each case consisting of $M = 10$ agents. The local field of view is set as $r_w \times r_h = 7 \times 7$. In order to fairly compare the performance of our approach with other existing methods and to assess the generalization capabilities, we generate 1000 test cases based on each predefined map as shown in Table 1. The robot density of predefined map is calculated by $\rho_r = M/(l_w * l_h)$. The generalization capability of the algorithm is evaluated on maps with robot density same as the training set, as well as on maps with varying robot density.

TABLE 1: Predefined maps in the test cases. $l_w \times l_h$ is the size of a map, and M is the number of robots on the map. In each map, the obstacle density (i.e., the ratio between the number of obstacles and total cells on a map) is set to 0.1.

Maps in Category 1 (same robot density)		Maps in Category 2 (varying robot densities)	
$l_w \times l_h$	M	$l_w \times l_h$	M
20x20	10	50x50	10
28x28	20	50x50	20
35x35	30	50x50	30
40x40	40	50x50	40
45x45	50	50x50	50
65x65	100	50x50	60

5.2 Performance Evaluation Metrics

In the experiments, we evaluate the performance of the proposed approach with three metrics, namely success rate, flowtime variation, and communication frequency.

(1) Success rate. It measures the ability of the algorithm to complete MAPP within a given time steps. It is defined as $s_r = n_{success}/n_{total}$, the proportion of successful cases $n_{success}$ over the number of total tested cases n_{total}. A higher success rate indicates that the algorithm is more successful in achieving planning completion in a timely manner.

(2) Flowtime variation. It quantifies the deviation of the planning completion time between learning-based methods and expert algorithms. It is described as $f_v = (\hat{E}_p - E_p^*)/E_p^*$. E_p is defined as the planning completion time taken for all agents to reach their respective goals. A lower value of f_v indicates greater resemblance to expert algorithms and superior performance.

(3) Communication frequency c_f. It characterizes the frequency of feature interaction in the information aggregation layer, reflecting the cost associated with communication. The lower c_f observed under the same success rate and flowtime variation indicates higher communication efficiency and better performance of the algorithm.

5.3 Compared Methods

We evaluate and compare the performance of our proposed approach GAT-GS (GS means global static feature extraction and selective communication) with several ablation methods including:

(1) GAT-S: without global static feature extraction.

(2) GAT-G: without selective communication. The central agent aggregates features from all agents within its field of view (FOV).

(3) GAT-GS-noS: without the strategy switching mechanism.

Additionally, we compare our methods with the state-of-the art imitation learning-based MAPP methods, GNN [7] and MAGAT-B [14]. To ensure a fair comparison, we set the planning time limit to three times that of the expert algorithm, and consider a case as failed if it cannot find the solution within the time limit. Figure 6 shows two examples of the simulated grid maps.

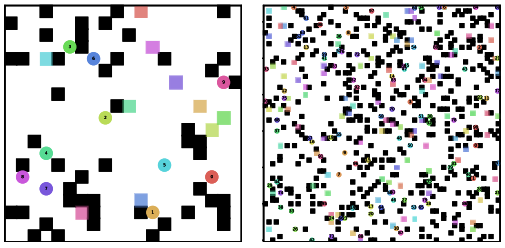

FIGURE 6: Two examples of the simulated grid maps. (a) 20×20 map size, 10 agents; (b) 65×65 map size, 100 agents.

5.4 Experiment Results

5.4.1 Success Rate and Flowtime Variation

Figure 7(a) shows that our proposed approach GAT-GS outperforms other methods in success rate. It has a consistently high success rate of approximately 95% in the maps with same robot density, which is the highest among all methods. Moreover, GAT-GS also shows the best performance in flowtime variation, with values consistently below 0.065 as shown in Figure 7(c). Even in the map with the highest robot density, GAT-GS achieves a success rate of over 93% (see Figure 7(b)), with a flowtime variation maintained around 0.1, which is the best performance among all methods. These results indicate the significant advantages of GAT-GS over other imitative learning-based algorithms such as GNN and MAGAT-B, particularly in terms of success rate.

Furthermore, the ablation experiments provide valuable insights into the performance of our approach. A comparison between GAT-GS and GAT-S shows that the inclusion of global static features in GAT-GS resulted in a significant improvement in the success rate, suggesting that incorporating additional reference information for local observation can contribute to higher success rate. A comparison among GAT-G and GAT-GS reveals that focusing on specific valuable neighboring agents rather than all agents in the field of view for feature aggregation can achieve better results, particularly in maps with high density of robots. This observation will be discussed in detail in the next subsection. Furthermore, our strategy switching mechanism shows promising results in improving the success rate, particularly in maps with low robot density. This indicates that the developed strategy switching mechanism has the potential to address the local trap problem to some extent.

5.4.2 Communication Frequency

Table 2 shows the communication frequency of two methods in maps with varying robot densities. We can see that the proposed GAT-GS approach can reduce communication frequency by an average of 70% compared to GAT-G These findings emphasize the importance of actively selecting appropriate agents for communication in decentralized multi-agent path planning. Lowering communication frequency can potentially reduce the chances of communication errors, packet loss, or interference, especially in real-world scenarios where wireless communications can be unpredictable. This makes GAT-GS a potentially more reliable method in challenging communication environments.

The results in Table 2 are consistent with the results in Figure 7. The proposed approach GAT-GS using the request-reply based selective communication procedure achieves the best results, while the GAT-G (the central agent communicates with all surrounding agents) has the worst performance. The findings from Figure 7 and Table 2 also indicate that increasing the frequency of communication interactions with surrounding agents does not necessarily result in improved performance of path planning.

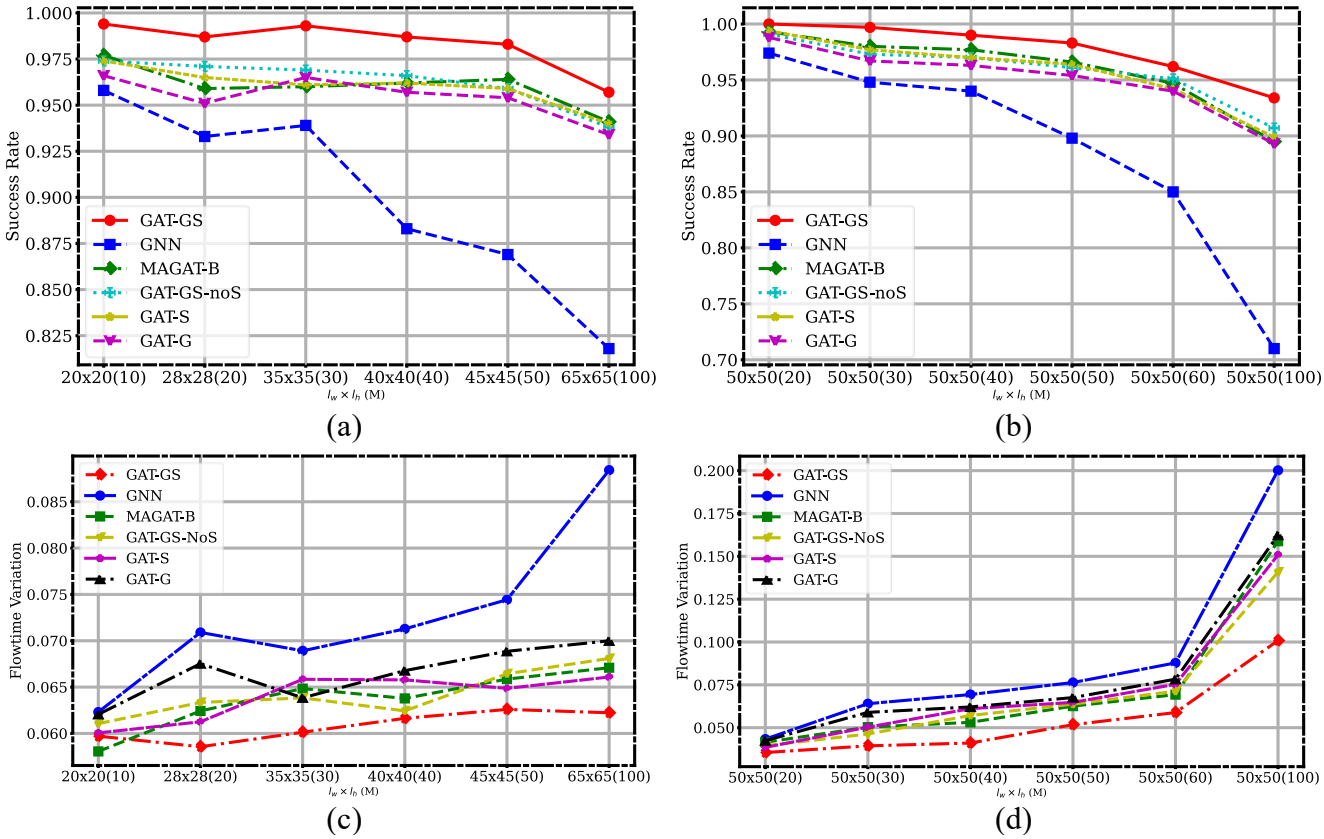

FIGURE 7: The success rate and flowtime variation of different methods against different maps. (a) and (c) show the results in maps with same robot density, while (b) and (d) show the results in maps with varying robot densities.

TABLE 2: Communication frequency of two methods in maps with varying robot densities. $l_w \times l_h$ is the size of a map, and M is the number of robots in the map.

$l_w \times l_h = 50 \times 50$		
M	GAT-GS	GAT-G
20	1155.3	4860.9
30	2224.6	8139.3
40	3835.9	12141.1
50	5742.2	16624.6
60	8217.4	21723.4
100	18397.1	45834.7

5.4.3 Discussion

The purpose of our experiments is to evaluate the effect of introducing the global static feature extraction, the request-reply based selective communication, and the strategy switching mechanism in improving the performance of MAPP in different maps. Our findings show that these treatments indeed improve the algorithm performance to a certain extent. Additionally, our proposed approach exhibits similar performance to expert algorithms in terms of flowtime variation.

One limitation of our approach is the additional computational load when computing the A* expert algorithm under a global static map, especially when dealing with a large number of agents. However, since the planning process only considers the static map and the robot's own location information, the computational burden remains manageable and can be distributed across multiple robots. Furthermore, the paths calculated from the A* expert algorithm serve as a guide for later selective communication procedure, which can reduce the communication load between agents greatly. This is especially important in high-density or highly dynamic environments, where robot-to-robot communication may be subject to interference or connection loss, resulting in failed path planning.

6. CONCLUSION

In this paper, we present a decentralized MAPP approach based on imitation learning and global static feature extraction. Our approach incorporates global static feature information to augment the available information sources during decentralized path planning. We also design a request-reply based selective communication procedure that allows agents to proactively choose relevant and impactful neighbors for communication. Additionally, we develop a strategy switching mechanism that adaptively utilizes expert-planned paths under a global static map to assist agents in escaping from local traps. The effectiveness of our proposed approach is evaluated through extensive simulations in grid environments with varying map sizes and numbers of agents. The experimental results demonstrate that our approach achieves a high success rate in path planning while significantly reducing the communication load. Our approach can be particularly advantageous in scenarios with low communication bandwidth or unstable communication quality. Overall, this study offers a promising solution for decentralized MAPP, and we expect that our work will inspire the development of more advanced methods in this area.

For future research directions, we recommend further investigation into improving the generalizability and robustness of our approach in more complex and dynamic environments, as well as exploring potential extensions of our approach to real-world implementations. Meanwhile, we intend to design various network architectures to further improve the performance of our approach. Additionally, the combination of our approach with other machine learning techniques, such as reinforcement learning or meta-learning, could also be an interesting direction for future work.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support from National Key R&D Program of China (2022YFB4702400).

REFERENCES

[1] De Ryck, M., Versteyhe, M., and Debrouwere, F., 2020, "Automated Guided Vehicle Systems, State-of-the-Art Control Algorithms and Techniques," J Manuf Syst, **54**, pp. 152–173.

[2] Jafari, N., Azarian, M., and Yu, H., 2022, "Moving from Industry 4.0 to Industry 5.0: What Are the Implications for Smart Logistics?," Logistics, **6**(2), p. 26.

[3] Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T. K., Barták, R., and Boyarski, E., 2021, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," Proceedings of the International Symposium on Combinatorial Search, **10**(1), pp. 151–158.

[4] Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R., 2015, "Conflict-Based Search for Optimal Multi-Agent Pathfinding," Artif Intell, **219**, pp. 40–66.

[5] Wagner, G., and Choset, H., 2011, "M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 3260–3267.

[6] Sartoretti, G., Kerr, J., Shi, Y., Wagner, G., Kumar, T. K. S., Koenig, S., and Choset, H., 2019, "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning," IEEE Robot Autom Lett, **4**(3), pp. 2378–2385.

[7] Li, Q., Gama, F., Ribeiro, A., and Prorok, A., 2020, "Graph Neural Networks for Decentralized Multi-Robot Path Planning," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 11785–11792.

[8] Barer, M., Sharon, G., Stern, R., and Felner, A., 2021, "Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem," Proceedings of the International Symposium on Combinatorial Search, **5**(1), pp. 19–27.

[9] Ferner, C., Wagner, G., and Choset, H., 2013, "ODrM* Optimal Multirobot Path Planning in Low Dimensional Search Spaces," *2013 IEEE International Conference on Robotics and Automation*, IEEE, pp. 3854–3859.

[10] van den Berg, J., Guy, S. J., Lin, M., and Manocha, D., 2011, "Reciprocal N-Body Collision Avoidance," In Robotics Research: The 14th International Symposium ISRR, pp. 3–19.

[11] Prorok, A., Blumenkamp, J., Li, Q., Kortvelesy, R., Liu, Z., and Stump, E., 2022, "The Holy Grail of Multi-Robot Planning: Learning to Generate Online-Scalable Solutions from Offline-Optimal Experts," *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1804–1808.

[12] Ma, Z., Luo, Y., and Ma, H., 2021, "Distributed Heuristic Multi-Agent Path Finding with Communication," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 8699–8705.

[13] Chen, L., Wang, Y., Miao, Z., Mo, Y., Feng, M., Zhou, Z., and Wang, H., 2023, "Transformer-Based Imitative Reinforcement Learning for Multi-Robot Path Planning," IEEE Trans Industr Inform.

[14] Li, Q., Lin, W., Liu, Z., and Prorok, A., 2021, "Message-Aware Graph Attention Networks for Large-Scale Multi-Robot Path Planning," IEEE Robot Autom Lett, **6**(3), pp. 5533–5540.

[15] Zhang, S. Q., Lin, J., and Zhang, Q., 2020, "Succinct and Robust Multi-Agent Communication with Temporal Message Control," *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA.

[16] Ding, Z., Huang, T., and Lu, Z., 2020, "Learning Individually Inferred Communication for Multi-Agent Cooperation," *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA.

[17] Ma, Z., Luo, Y., and Pan, J., 2022, "Learning Selective Communication for Multi-Agent Path Finding," IEEE Robot Autom Lett, **7**(2), pp. 1455–1462.

[18] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y., 2017, "Graph Attention Networks."