ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

A Decentralized Multi-Agent Path Planning Approach Based on Imitation Learning and Selective Communication

Bohan Feng
University of Michigan—Shanghai Jiao Tong University Joint Institute,
Shanghai Jiao Tong University,
800 Dongchuan Road, Minhang District,
Shanghai 200240, China
e-mail: bohan.feng@sjtu.edu.cn

Youyi Bi[1]
University of Michigan—Shanghai Jiao Tong University Joint Institute,
Shanghai Jiao Tong University,
800 Dongchuan Road, Minhang District,
Shanghai 200240, China
e-mail: youyi.bi@sjtu.edu.cn

Mian Li
Global Institute of Future Technology,
Shanghai Jiao Tong University,
800 Dongchuan Road, Minhang District,
Shanghai 200240, China
e-mail: mianli@sjtu.edu.cn

Liyong Lin
Contemporary Amperex Technology Co., Limited,
2 Xingang Road, Zhangwan Town,
Jiaocheng District,
Ningde, Fujian 352100, China
e-mail: llin5@e.ntu.edu.sg

Multi-agent path planning (MAPP) is crucial for large-scale mobile robot systems to work safely and properly in complex environments. Existing learning-based decentralized MAPP approaches allow each agent to gather information from nearby agents, leading to more efficient coordination among agents. However, these approaches often struggle with reasonably handling local information inputs for each agent, and their communication mechanisms between agents need to be further refined to treat those congested traffic scenarios effectively. To address these issues, we propose a decentralized MAPP approach based on imitation learning and selective communication. Our approach adopts an imitation learning architecture that enables agents to rapidly learn complex behaviors from expert planning experience. The information extraction layer is integrated with convolutional neural network (CNN) and gated recurrent unit (GRU) for capturing features from local field-of-view observations. A two-stage selective communication process based on graph attention neural network (GAT) is developed to reduce the required neighbor agents in inter-agent communication. In addition, an adaptive strategy switching mechanism utilizing local expert-planned paths is designed to support robots 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 show that our approach outperforms other decentralized path planning methods in success rate while maintaining the lowest flowtime variation and communication frequency. Furthermore, our approach is computationally efficient and scalable, making it suitable for real-world applications.
[DOI: 10.1115/1.4065167]

Keywords: decentralized control, multi-agent path planning, mobile robot, selective communication, machine learning for engineering applications

1 Introduction

In the era of Industry 4.0, mobile robots are playing a significant role in revolutionizing diverse facets of contemporary production paradigms [1–3]. Consequently, the development of efficient multi-agent path planning (MAPP) algorithm is of prominent importance. Traditional MAPP typically adopts a centralized paradigm, in which a central planning unit generates paths for all agents. Centralized approaches can offer optimality and completeness in their solutions, ensuring that the optimal path is found for each agent if it exists [4,5]. However, the scalability of these approaches is insufficient as the number of robots increases, making them less suitable for complex environments with large-scale robot systems [6].

Recent studies are exploring decentralized MAPP in which each agent computes its own path based on local information [7,8]. Decentralized paradigm significantly alleviates the computational burden of the central planning unit, making it more scalable for large-scale systems. Nevertheless, in practice, decentralized approaches often achieve limited successes in complex environments. In addition, they may fail to consistently produce optimal solutions, as they focus on generating paths for individual agents based on local information rather than considering the overall system performance. Moreover, decentralized approaches can potentially cause excessive and unnecessary communication among agents, due to the need of frequent exchanging of local information during path planning.

Therefore, we propose a novel decentralized MAPP approach based on imitation learning and selective communication (DILSC) to enhance the practicality of decentralized path planning. The effectiveness of the proposed approach is validated in both simulations and physical experiments. The primary contribution of this study includes:

- A new information extraction layer for imitation learning (IL)-based decentralized MAPP is designed. This layer innovatively combines convolutional neural network (CNN) and gated recurrent unit (GRU) to extract critical feature from local observation.
- A two-stage selective communication process for decentralized MAPP is proposed. This process enables agents to interact efficiently with selected influential neighbors via graph attention network (GAT).
- A strategy switching mechanism to assist agents navigating out of local traps is developed. This mechanism adaptively

[1]Corresponding author.
Manuscript received July 22, 2023; final manuscript received March 17, 2024; published online April 16, 2024. Assoc. Editor: Atul Thakur.

utilizes local expert planning method or learning-based algorithm to generate paths.

2 Related Work

MAPP methods can be classified into centralized and decentralized ones. In centralized methods, a central planning unit calculates coordinated paths for all agents. Since all agents share a common state space, centralized methods can provide optimal and complete path plans, but their computational demand is high when the working environment is complex. For example, conflict-based search (CBS) and its variant enhanced CBS (ECBS) [4,9] can find optimal or suboptimal path solutions, where the high-level central unit searches for a set of collision constraints and imposes 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 collisions occur. Although some progress has been made in reducing the computational load, these centralized methods still rely on a central unit to compute all paths, and thus struggle to handle environments with numerous potential path conflicts.

In decentralized MAPP methods, agents make independent decisions based on local environment information [10]. In recent years, learning-based decentralized MAPP algorithms have received increasing interest. Researchers explored to use imitation learning or reinforcement learning (RL) to train agents to move independently according to locally sensed information [11,12]. For example, PRIMAL (pathfinding via reinforcement and imitation multi-agent learning) [7] is a classical framework that demonstrates the feasibility of learning-based decentralized MAPP. Graph neural network (GNN)-based models are gaining prominence for their adaptability of managing complex interactions in multi-agent systems. Li et al. [8,13] explore the use of GNN for explicit communication in complex multi-agent coordination, utilizing imitation learning to approximate centralized expert algorithms. Kool et al. [14] focus on attention mechanism for routing optimization, while Paul et al. [15] integrate capsule network with multi-head attention for task allocation. Additional research advancements include Wang et al.'s [16] heterogeneous graph attention network and Paul et al.'s [17] graph reinforcement learning method with higher order topological abstraction. Distributed, heuristic and communication [18] combines graph neural network with deep Q-learning to improve policy performance in complex-obstacle environments. These studies underscore the importance of attention mechanism, communication efficiency, and dynamic adaptability in complex robotic environments, offering crucial insights for developing more efficient MAPP methodology. However, current learning-based MAPP algorithms have not adequately addressed the following important issues:

- Decentralized MAPP algorithms leverage feature extraction to obtain critical characteristics of the local environment, thereby assisting in individual path planning. However, the identification of the most crucial features for decentralized path planning and the design of proper feature extraction network remain elusive and demand further exploration.
- In decentralized MAPP, frequent communications among agents lead to high communication and computational burden. Researchers have proposed several methods to alleviate this issue, such as attention mechanism [13], temporal message control [19], individually inferred communication (I2C) [20], decision causal communication [21], learning structured communication [22], etc. While some of these methods can adjust information passing among agents, they may struggle to effectively deal with redundant or repetitive data that do not offer new insights into the path planning process. It is still not fully clear how the communication between neighbors can be further reduced according to the local feature information.
- Although existing learning-based MAPP algorithms have achieved good performance in many scenarios, it is still challenging for them to handle local traps effectively [23], in which an agent oscillates around a few positions as its outward passages are blocked by obstacles or other agents.

In this study, we expect to address the above issues by introducing a decentralized MAPP approach based on imitation learning and selective communication.

2.1 Problem Formulation.

Our research focus is placed on 2D grid maps, characterized by four-neighbor connectivity. Each entity (i.e., an agent or an obstacle) occupies a single grid cell. The map is represented by an $l_w \times l_h \in \mathbb{R}^{l_w \times l_h}$ graph matrix. For each map, we choose the starting and corresponding goal vertices for M agents from the available free positions.

We consider a partially observable discrete grid world, where agents can only observe the state of the world in a limited field-of-view (FOV) $r_w \times r_h \in \mathbb{R}^{r_w \times r_h}$ centered around themselves. The decision-making process of agents is synchronous, operating within a discretized time framework. At each time-step, agent can either move to an adjacent vertex or wait at its current vertex, resulting in an action space with a size of five (i.e., move upward, downward, left, right, or remain stationary). Each agent can communicate or share information with only adjacent agents within its FOV. Compared with the time for the agent's movement, the communication between the neighboring agents happens instantly with negligible delay, and is not blocked by any obstacles. Both the time cost in move and wait actions will be counted unless the agent waits at its goal vertex without new assigned tasks. A tuple $\langle i, j, v, t \rangle$ is used to describe a vertex conflict, where agents i and j are at the same vertex v at time-step t. A tuple $\langle i, j, u, v, t \rangle$ represents an edge conflict, where agents i and j traverse the same edge (u, v) in opposite directions between time-steps t and $t+1$. The overall objective is to find a set of conflict-free paths which move all agents from their start vertices to goal vertices while minimizing the maximal completion time of these paths. The objective function for this scenario can be formulated as Eq. (1):

$$\min \max_{i \in \{1,\ldots,M\}} (T_i) \tag{1}$$

where T_i represents the completion time for agent i to reach its goal vertex.

3 Methods

3.1 The Overall Structure of the Proposed Approach.

Figure 1 shows the overall workflow of our approach, which adopts an imitation learning architecture consisting of three layers: information extraction, information aggregation, and action output. The observation of agent i within its FOV is first processed through the local observation processing to get o_i^t. The agent i's local observation is updated at each time-step. The local observation o_i^t is first encoded into $\tilde{o}_i^t \in \mathbb{R}^H$ by the CNN module. Then by adopting the last time-step aggregation outcome $\tilde{h}_i^{t-1} \in \mathbb{R}^H$ as the hidden state and $\tilde{o}_i^t$, the GRU module generates $\hat{o}_i^t \in \mathbb{R}^H$. Subsequently, $\hat{o}_i^t$ is passed into the information aggregation layer, which aggregates information through the two-stage selective communication process and graph attention network to obtain $\tilde{h}_i^t \in \mathbb{R}^H$. Finally, $\tilde{h}_i^t$ 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 and the learning-based algorithm to help agents escape from local traps. In the following subsections, the detailed explanations of our approach are provided.

3.2 Local Observation Process.

We denote the observation of agent i within its FOV through the local observation process as o_i^t. As illustrated in Fig. 2, the available information in the finite FOV

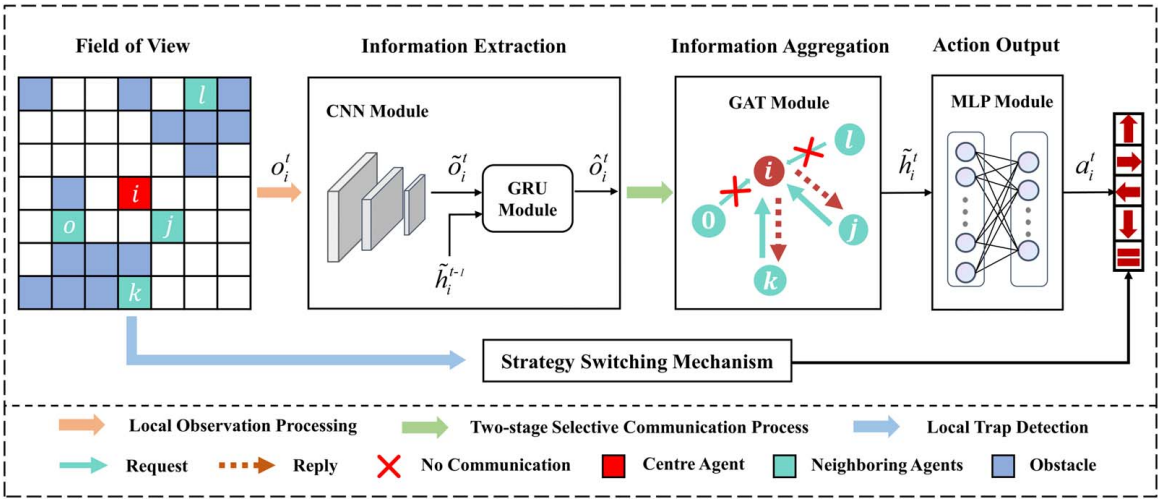

Fig. 1 The overall workflow of the proposed decentralized MAPP approach

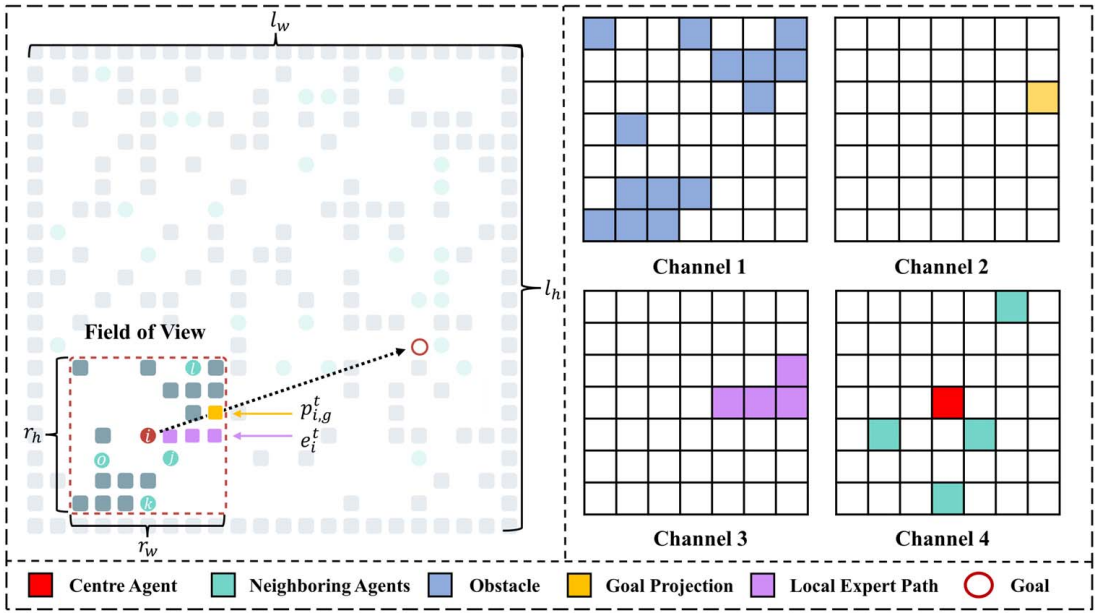

Fig. 2 The available information in the finite FOV for path planning (i.e., four channels)

for path planning is divided into four distinct channels to aid learning. Specifically, channel 1 contains the obstacles in the agent i's FOV. Channel 2 represents the position of goal $v_{i,g}$ or its projection into the boundary of FOV $p^t_{i,g}$. Channel 3 displays the expert path e^t_i from the location of agent i to $p^t_{i,g}$ calculated by A* algorithm. Channel 4 shows the location of agent i and 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 observations with higher efficiency.

3.3 Information Extraction Layer. Our information extraction layer leverages the robust feature extraction capabilities of CNN [24] and the temporal sequence capturing ability of GRU [25]. As depicted in Fig. 3, the input observation o^t_i is processed by a CNN operating internally on agent i. Within this CNN architecture, each of the three modules comprises a sequence of operations: Conv2D, followed by BatchNorm2D, ReLU activation, and MaxPooling. CNN generates a vector $\tilde{o}^t_i \in \mathbb{R}^H$, where H refers to the dimensionality of the output vector ($\tilde{o}^t_i = \text{CNN}(o^t_i)$). Then $\tilde{o}^t_i$ serves as the input state for the GRU, while $\hat{h}^{t-1}_i$ is employed as its corresponding hidden state. Crucially, $\hat{h}^{t-1}_i$ represents the aggregated feature extracted from the information aggregation layer at the preceding time-step. For the GRU, the reset gate $R^t \in \mathbb{R}^H$ and update gate $Z^t \in \mathbb{R}^H$ are computed as follows:

$$R^t = \sigma(\tilde{o}^t_i W_{or} + \hat{h}^{t-1}_i W_{hr} + b_r) \tag{2}$$

$$Z^t = \sigma(\tilde{o}^t_i W_{oz} + \hat{h}^{t-1}_i W_{hz} + b_z) \tag{3}$$

where $W_{or}, W_{oz} \in \mathbb{R}^{H\times H}$ and $W_{hr}, W_{hz} \in \mathbb{R}^{H\times H}$ are weight parameters and $b_r, b_z \in \mathbb{R}^H$ are bias parameters. Next, we integrate the reset gate R^t with the updating mechanism, leading to the following candidate hidden state $\bar{h}^t_i$ at time-step t:

$$\bar{h}^t_i = \tanh(\tilde{o}^t_i W_{oh} + (R^t \odot \hat{h}^{t-1}_i) W_{hh} + b_h) \tag{4}$$

where $W_{oh} \in \mathbb{R}^{H\times H}$ and $W_{hh} \in \mathbb{R}^{H\times H}$ are weight parameters, $b_z \in \mathbb{R}^H$ is the bias, and the symbol $\odot$ is the Hadamard product operator. Also, we use tanh activation function.

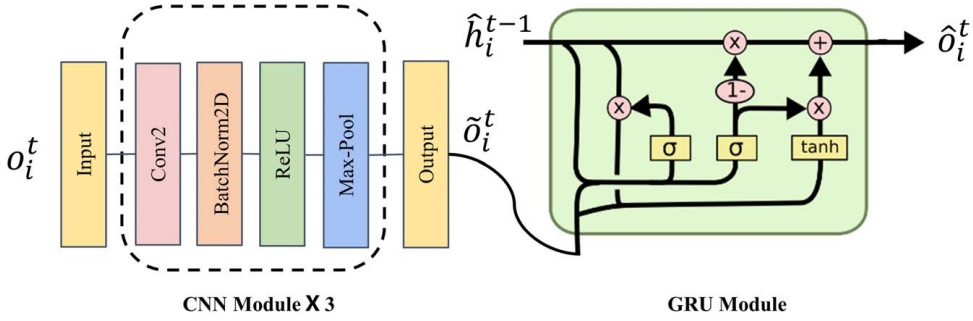

Fig. 3 The CNN module and GRU module for information extraction

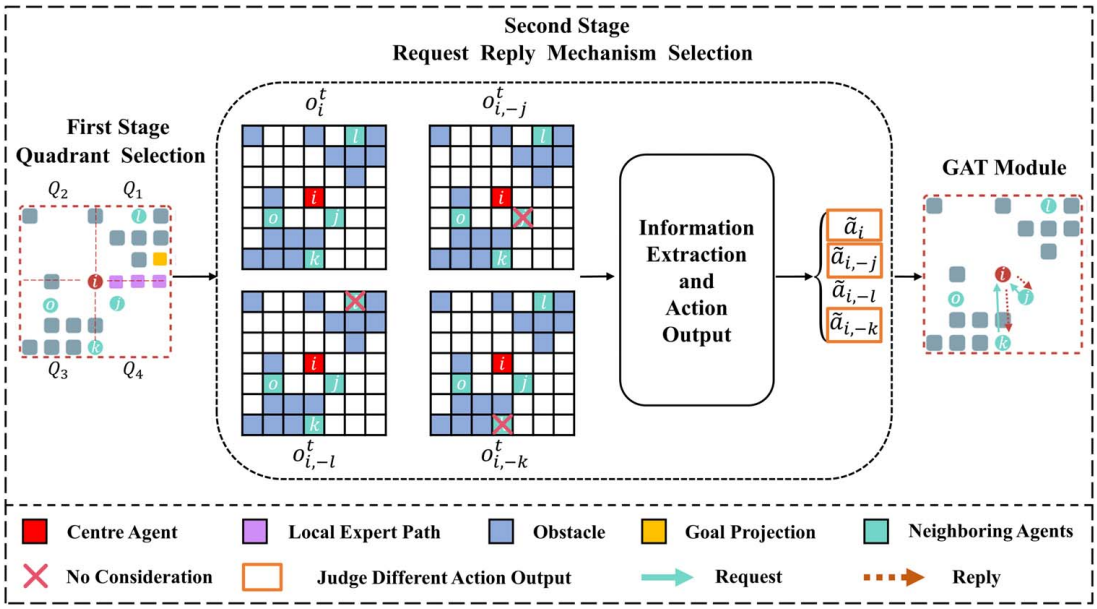

Fig. 4 An illustration of the two-stage selective communication process

Finally, we incorporate the effect of the updating gate Z^t and get the final update equation for the GRU:

$$\hat{o}_i^t = \mathrm{GRU}(o_i^t, \hat{h}_i^{t-1}) = Z^t \odot \hat{h}_i^{t-1} + (1 - Z^t) \odot \bar{h}_i^t \tag{5}$$

The generated $\hat{o}_i^t$ will undergo further processing through information aggregation layer as detailed in the subsequent section.

3.4 Information Aggregation Layer. The information aggregation layer comprises two parts: the two-stage selective communication process and the implementation of a graph attention network for feature aggregation with the selected agents.

3.4.1 The Two-Stage Selective Communication Process. Figure 4 illustrates the two-stage selective communication process. The first stage is quadrant selection based on local observation information. Let $Q_i = \{Q_1, Q_2, Q_3, Q_4\}$ be the four quadrants of agent i's observation. We can then determine a smaller quadrant set Q_i', such as Q_1 and Q_4 illustrated in Fig. 4, 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 FOV and e_i^t is the local expert-planned path. 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' (agents j, k, and l in Fig. 4) as the set of agents located in Q_i'.

The second stage 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 output $o_{i,-j}^t$ ($-j$ 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 $\hat{o}_{i,-j}^t$. By skipping the information aggregation layer, $\hat{o}_i^t$ and $\hat{o}_{i,-j}^i$ 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 with agent j. Equation (5) shows the selective communication function for agent i:

$$\rho_{ij} = \begin{cases} 0 & \text{if} \quad \tilde{a}_i^t = \tilde{a}_{i,-j}^t,\ i \in M,\ j \in N_i' \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

Note that temporary actions $\tilde{a}_i^t$ and $\tilde{a}_{i,-j}^t$ are only used to determine 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 $\mathcal{S}_i$ (j and k agents in Fig. 4) as shown in Eq. (6):

$$\mathcal{S}_i = \{ j | \rho_{ij} = 1,\ j \in N_i' \} \tag{7}$$

3.4.2 The Graph Attention Network. After getting the neighboring agents with necessity of communication through the selective communication, we use GAT [26] to aggregate the feature

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

$$h_i^t = W(\hat{o}_i^t)^T \tag{8}$$

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

$$e_{ij}^t = \text{attention}(h_i^t, h_j^t) = \text{LeakyReLU}(\beta^T [h_i^t \oplus h_j^t]) \tag{9}$$

where $\beta \in \mathbb{R}^{2H'}$ 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:

$$\alpha_{ij}^t = \text{softmax}(e_{ij}^t) = \frac{\exp(e_{ij}^t)}{\sum_{k \in \mathcal{S}(i)} \exp(e_{ik}^t)} \tag{10}$$

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

$$\tilde{h}_i^t = \sigma\left(\sum_{j \in \mathcal{S}(i)} \alpha_{ij}^t h_j^t\right) \tag{11}$$

where σ: R $\to$ R denotes a sigmoid activation function. To improve the model's capacity and stability, we use multi-head attention [27], where we learn K independent attention mechanisms:

$$\tilde{h}_i^{t(k)} = \sigma\left(\sum_{j \in \mathcal{S}(i)} \alpha_{ij}^{t(k)} W^k (\hat{o}_i^t)^T\right) \tag{12}$$

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

$$\hat{h}_i^t = f\left(\oplus_{k=1}^K \tilde{h}_i^{t(k)}\right) \tag{13}$$

where $\oplus$ represents concatenation and f is one neural network layer. Note that, in this setting, the final returned output, $\hat{h}_i^t$, will consist of H features for each node.

3.5 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., $a_i^t = \text{MLP}(\hat{h}_i^t)$. A softmax function is applied after the MLP layer to convert the action vector into a probability distribution vector for five discrete actions. The argmax function is used to choose the action with the highest probability, i.e., the final output action. 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 movement from the starting position to the goal position.

3.6 Training Procedure. 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. ECBS is guaranteed to find solutions whose costs are no more than a user-specified factor ω_E away from optimal. The methodology of our imitation learning approach specifically utilizes behavior cloning, a straightforward yet effective strategy where the agent's policy is directly learned by mimicking the expert's actions in similar states. At the beginning of training, we have the expert actions $\{E^t\}$ for all agents and the corresponding local observation $\{O^t\}$, collected in the training set $\mathcal{T} = \{(\{E^t\}, \{O^t\})\}$. Then, we use the cross-entropy loss function $\mathcal{L}$ to train the network function approximator $F(O^t; \theta)$:

$$L = -\frac{1}{M} \sum_{t=1}^{T_i} \sum_{i=1}^{M} E_i^t \log F(o_i^t; \theta) \tag{14}$$

where M is the count of agents, T_i represents the completion time-step for agent i, and θ denotes the parameter of the function approximator F.

Moreover, to refine our training dataset and improve the performance of the learned policy, we employ the DAgger (dataset aggregation) technique [28], which iteratively updates the training set by aggregating data from both the expert's policy and the learned policy. For every κ 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, which augments its diversity with an expanded range of examples and enhances its comprehensiveness by incorporating a wider spectrum of scenarios that the model might face. 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, thereby increasing decision-making quality and mitigating future path deviations.

3.7 The Strategy Switching Mechanism. To mitigate local trap issue and increase the success rate, we develop a strategy switching mechanism. As depicted in Fig. 5, the mechanism operates as follows:

(1) Detection of local traps: We first design a local trap detector that monitors the states of agents and identifies the trapping situations. We create a visit counter matrix C and each element in this matrix corresponds to a grid cell and stores the number of times that the agent has visited to that cell. 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 the agent i falling into local trap.
(2) Switching to expert guidance: Once local trap is detected, the mechanism expands agent i's outer boundary of FOV by adding an additional grid-based square layer of blank cells. We get the new goal projection $\tilde{p}_{i,g}^t$ into the new outer boundary. Then the path planning strategy switches to expert-planed path guidance for the agent i from current location to $\tilde{p}_{i,g}^t$ (see the purple cells in Fig. 5) generated from the A* algorithm.
(3) Switching back to learning-based algorithm: If the agent has successfully completed the predetermined number of steps η or reaches the boundary of its FOV, the path planning strategy reverts to the learning-based path planning algorithm (see the green cells in Fig. 5).

4 Experiment Setting and Results

4.1 Model Parameters and Environment Settings. In the information extraction layer, we use a convolutional kernel size of 3, with a stride of 1 and no padding for the CNN module. The GRU module consists of one layer. Prior to the aggregation layer, we set the number of shared features (H) to 64. In the information aggregation layer, a graph attention network of one layer is employed, using a K value of 4 for the multi-head attention mechanism. During the network training phase, we perform a check every $\kappa = 4$ epochs by randomly selecting $n_{\text{random}} = 200$ cases from the training set, identifying any failed cases or those deviating significantly ($r_d = 1.2$) from expert-planned paths. The Adam optimizer, featuring a momentum of 0.9, is utilized. A dynamic learning

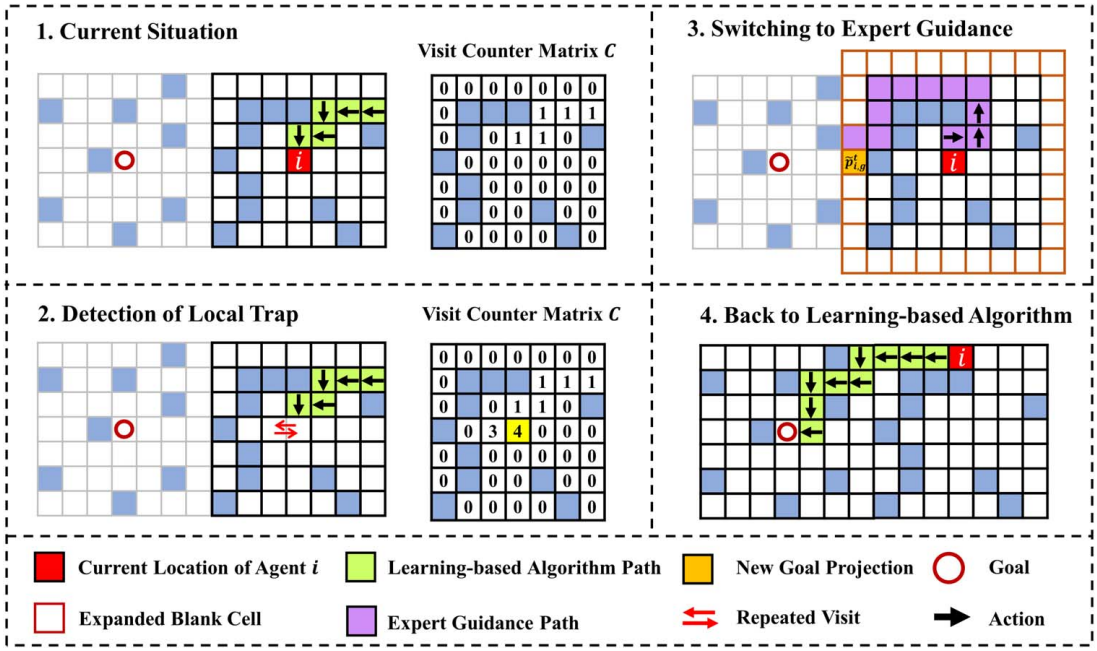

Fig. 5 An illustration of the strategy switching mechanism

Table 1 Predefined two categories of test case maps

Maps in category 1 (same robot density)		Maps in category 2 (varying robot density)	
$l_w \times l_h$	M	$l_w \times l_h$	M
20×20	10	50×50	10
28×28	20	50×50	20
35×35	30	50×50	30
40×40	40	50×50	40
45×45	50	50×50	50
65×65	100	50×50	60

Note: $l_w \times l_h$ is the size of a map, and M is the number of robots on the map. In each map, the obstacle density is set to 0.1.

rate is used with a starting value of 10^{-4}, which is decreased by 50% at 200 epochs and 400 epochs, respectively. For the inference phase, we set the visit count threshold (δ) at 4 and predetermine the number of steps (η) to r_w. We test our approach on a computer with Intel i7-10300 CPU, Nvidia RTX3090 GPU and 32 GB RAM.

According to the environment settings in previous research [8,13], we initialize 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 is also worth mentioning that each map generates 60 cases, with each case consisting of $M = 10$ agents. The optimality bound factor ω_E of ECBS is set to 1.1. The local FOV is set as $r_w \times r_h = 7 \times 7$. We generate 1000 test cases based on each predefined map as shown in Table 1. The robot density of predefined map is calculated by $\rho_r = M/(l_w * l_h)$. Figure 6 shows three examples of the simulated grid maps.

4.2 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-step. It is defined as $s_r = n_{\text{success}}/n_{\text{total}}$, the proportion of successful cases n_{success} over the number of total tested cases n_{total}.

(2) Flowtime variation. It quantifies the deviation of the planning completion time between learning-based methods and expert algorithms. It is described as $f_v = (L_p - E_p)/E_p$. Here, E_p represents the time taken by expert algorithms for all agents to reach their respective goals, while L_p denotes the time taken by learning-based algorithms for the same task. 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.

4.3 Compared Methods. Our study evaluates the DILSC against various ablation models:

(1) DILSC-G: This variant omits the GRU module, testing the impact of recurrent processing on performance.

(2) DILSC-S: This version excludes selective communication, with the central agent aggregating features from all agents within its FOV instead.

For a broader comparison, DILSC is also benchmarked against leading IL-based MAPP methods like GNN [8] and MAGAT (message-aware graph attention network) [13]. Both methods use convolutional neural network for feature extraction: GNN integrates features via graph neural network, while MAGAT leverages multi-head attention network for aggregation. 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. RL-based methods are not included in this comparison due to their inherent differences in learning paradigms and evaluation criteria as well as the complexities in ensuring fair comparisons between them and IL-based methods.

4.4 Experimental Results

4.4.1 Success Rate and Flowtime Variation. Figure 7(*a*) shows that our proposed approach DILSC outperforms other methods in success rate. It has a consistently high success rate of approximately

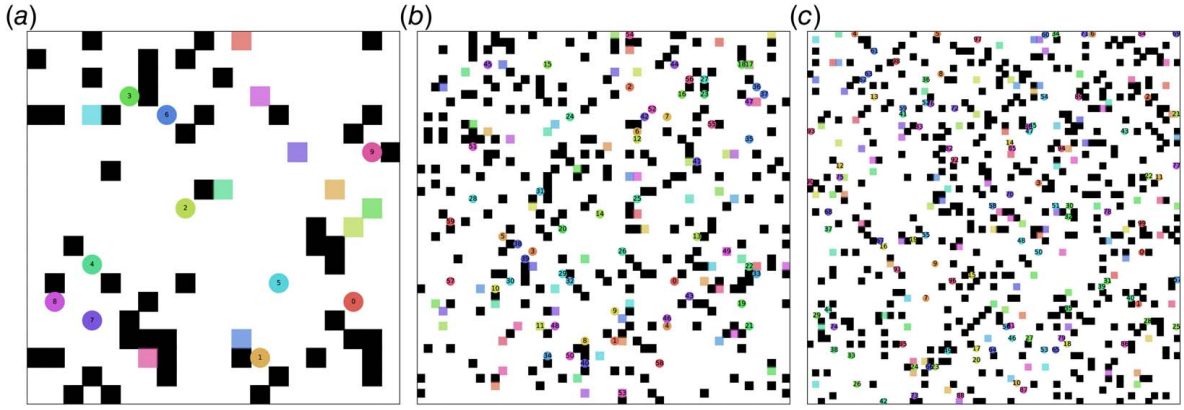

Fig. 6 Three examples of the simulated grid maps: (*a*) 20 × 20 map size, 10 agents; (*b*) 50 × 50 map size, 60 agents; and (*c*) 65 × 65 map size, 100 agents

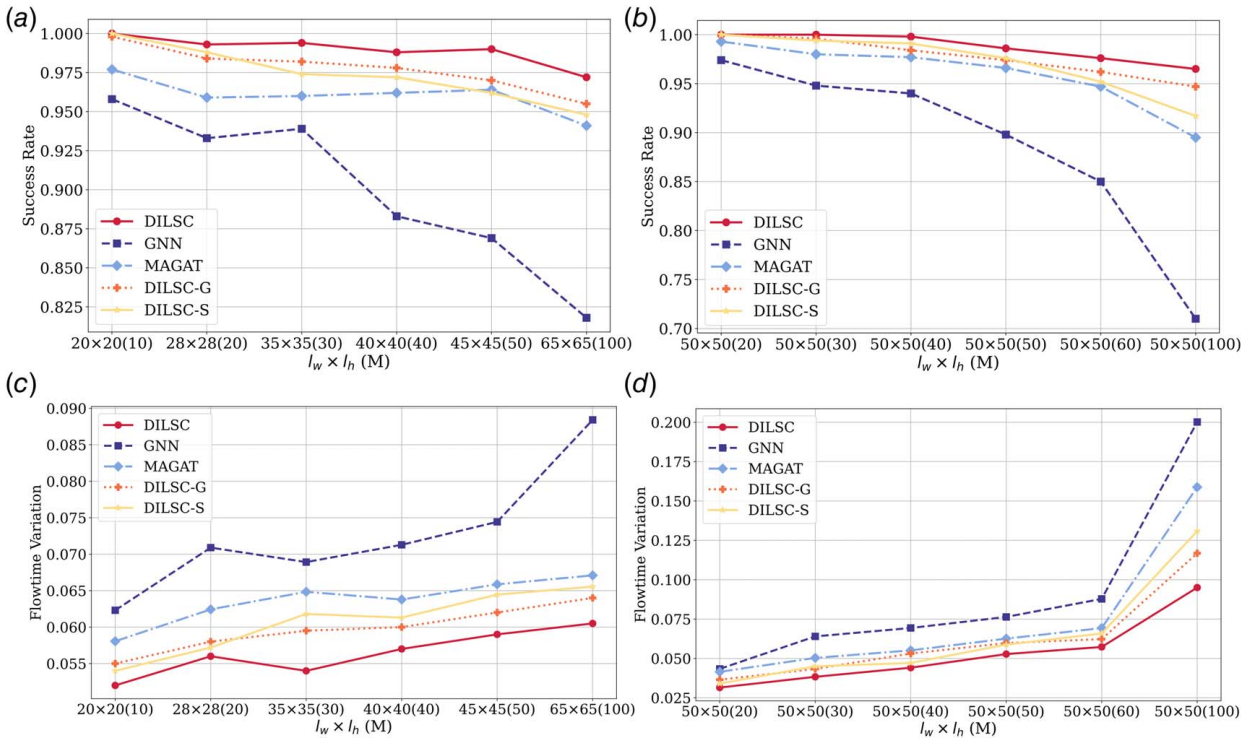

Fig. 7 The success rate and flowtime variation of different methods: (*a*) and (c) show the results in maps with same robot density, while (*b*) and (*d*) show the results in maps with varying robot density

97% in the maps with same robot density, which is the highest among all methods. Moreover, DILSC also shows the best performance in flowtime variation, with values consistently below 0.061 as shown in Fig. 7(*c*). Even in the map with varying robot density, DILSC achieves a success rate of over 96% (see Fig. 7(*b*)), with a flowtime variation maintained around 0.1 (see Fig. 7(*d*)), which is the best performance among all methods. These results markedly illustrate the distinct advantages that DILSC holds over other imitative learning-based algorithms, such as GNN and MAGAT, specifically in terms of success rate. Drawing insights from the flowtime variation results, it is noteworthy that our decentralized approach parallels the performance of the centralized expert algorithm closely, affirming the efficacy of our decentralized approach.

Furthermore, a comparison between DILSC and DILSC-G shows that the inclusion of GRU module in DILSC results in a significant improvement in the success rate (see Figs. 7(*a*) and 7(*b*)), suggesting that incorporating the ability to capture temporal dependencies in MAPP can contribute to higher success rate. A comparison between DILSC and DILSC-S reveals that focusing on specific valuable neighboring agents rather than all agents in the FOV for feature aggregation can achieve better results, particularly in maps with high density of robots.

4.4.2 Communication Frequency. Table 2 shows the average communication frequency of DILSC and DILSC-S. Remarkably, the proposed DILSC approach can reduce communication frequency by an average of 70% compared to DILSC-S in maps with same robot density 0.025. A closer look at maps sized 50 × 50 with 100 robots (belonging to category 2) reveals that even under the most extreme robot density (0.04) conditions, the efficiency of the DILSC approach still holds. It continues to reduce communication frequency, this time by approximately 58%. The results in Table 2 are consistent with the results in Fig. 7, implying that an increase in the frequency of communication interactions with surrounding agents does not necessarily enhance the

Table 2 Average communication frequency (c_f) of DILSC and DILSC-S

Maps in category 1 (same robot density)			Maps in category 2 (varying robot density)		
$l_w \times l_h(M)$	c_f of DILSC	c_f of DILSC-S	$l_w \times l_h(M)$	c_f of DILSC	c_f of DILSC-S
$20 \times 20(10)$	19.1	62.6	$50 \times 50(20)$	51.3	201.7
$28 \times 28(20)$	56.6	173.4	$50 \times 50(30)$	89.8	321.9
$35 \times 35(30)$	101.4	312.3	$50 \times 50(40)$	141.3	482.1
$40 \times 40(40)$	152.8	467.2	$50 \times 50(50)$	215.9	674.3
$45 \times 45(50)$	239.3	598.9	$50 \times 50(60)$	347.5	887.6
$65 \times 65(100)$	722.9	1815.7	$50 \times 50(100)$	651.2	1549.6

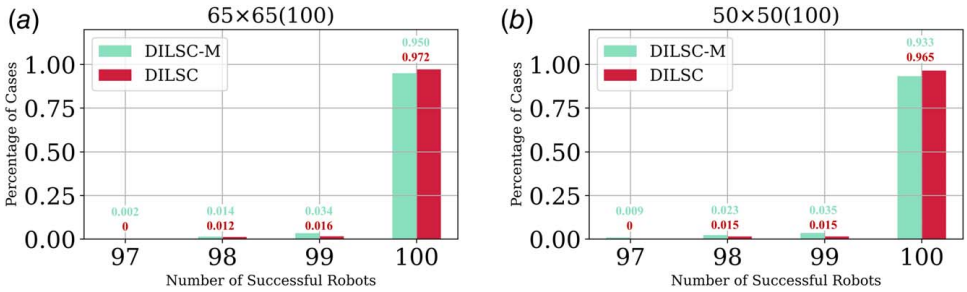

Fig. 8 Histogram of percentage of cases distributed over the number of successful agents: (*a*) shows the results in map size 65×65 with 100 robots, while (*b*) shows the results in map size 50×50 with 100 robots

performance of path planning. This result accentuates the necessity of active selection of appropriate agents for communication in decentralized MAPP.

4.4.3 Effect of Strategy Switching Mechanism. To examine the effect of the strategy switching mechanism, we further introduce a new model called DILSC-M lacking this mechanism. We compare the performance of DILSC and DILSC-M within two specific maps: 65×65 and 50×50 map both with 100 agents. Figure 8 provides a detailed histogram of the distribution of cases alongside the number of agents that successfully reach their destinations. Remarkably, the DILSC-M algorithm demonstrates that even in those unsuccessful cases, over 97% of the agents can arrive at their goals. A few robots encounter local traps, hindering their movement toward their goals. By comparing the distribution represented in the histograms of DILSC and DILSC-M, we find that the strategy switching is instrumental in helping trapped robots escape from predicaments, thereby ensuring their successful arrival at intended destinations.

4.4.4 Large-Scale Map Test. Compared to centralized MAPP methods, a major advantage of decentralized MAPP is its adaptivity to large-scale maps. To validate this advantage, we compare the performance of the proposed DILSC approach with centralized expert algorithm ECBS in large-scale map scenarios. The obstacle density of large-scale map is set to 0.1. Additionally, the optimality bound (denoted as ω_E) of ECBS is set to 3. Ten distinct cases of the large-scale map are generated. This test is a significant challenge since DILSC has been primarily trained on small 20×20 maps with a mere count of 10 agents.

Table 3 shows the results of large-scale map test. ECBS demonstrates difficulty when performs path planning for a significantly larger number of robots, 500 in this test. It requires averagely 6532 s in the computation, with a standard deviation of 1598 s, indicating a considerable performance variance across different cases. In contrast, DILSC is significantly more efficient, completing the planning in 601 s, or around one-tenth of the time consumed by ECBS. Moreover, DILSC demonstrates a smaller standard deviation of 114 s, implying a more reliable and consistent performance across varying test cases. Even though its initial training is based on small 20×20 maps with only 10 agents, it has demonstrated the capability to handle large-scale and complex environments. This suggests our approach's promising adaptability and potential applicability in handling even larger maps or more robots.

Table 3 Average time cost of ECBS and DILSC in the large-scale map test

$l_w \times l_h(M)$	ECBS time cost (std.)	DILSC time cost (std.)
$200 \times 200(500)$	6532 (1598)	601 (114)

Note: The unit used for time cost is second.

4.4.5 Real-World Validation Experiment. To demonstrate the feasibility of deploying our approach in a real-world environment, we test with three physical mobile robots. As depicted in Fig. 9, each robot is equipped with multiple sensors, including LiDAR (light detection and ranging), an inertial measurement unit, and a wheel tachometer. These sensors facilitate sensing environmental and robot's status information, providing each robot with self-localization ability. Communication between robots occurs via Wi-Fi, and the surrounding information of each robot is captured using LiDAR. The robot's control board transforms this information as the input of the proposed DILSC approach, and the action commands are issued to the robots' control board, driving their movements.

Our real-world experiment takes place in an 8×8 grid map, populated with eight obstacles and three mobile robots. We establish the robots' start positions, goals, and the obstacles' locations randomly for the experimental setup. Every robot employs our proposed approach to plan their subsequent actions. Notably, this computation is executed onboard using a Jetson Nano B01 on each robot, with each planning step completed in about 0.137 s. Figure 10 provides the snapshots of the robots' path planning in the real-world experiment. It starts with an image of the robots' initial status, followed by images showing the robots' movement through the task. Figure 10(*e*) displays the robots upon reaching their goals, with their individual paths delineated in different colors.

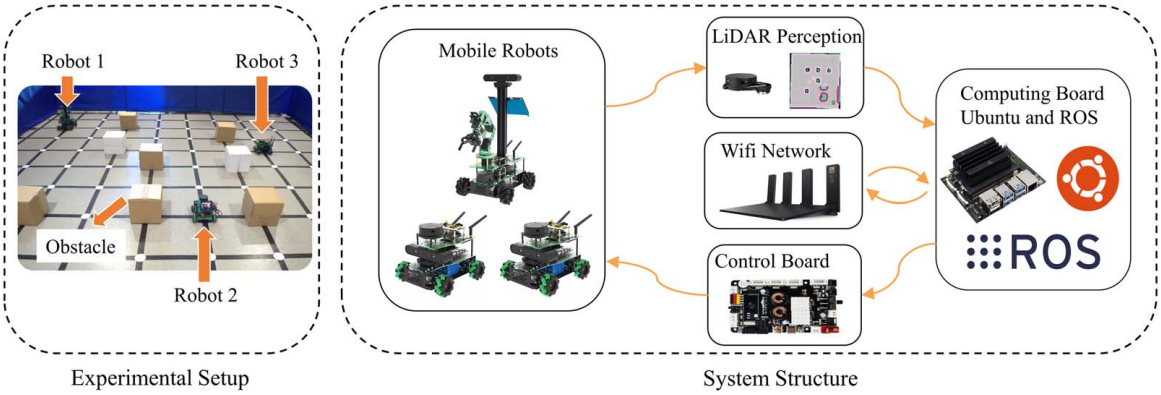

Fig. 9 Experimental setup and system structure of the mobile robots in a real-world environment

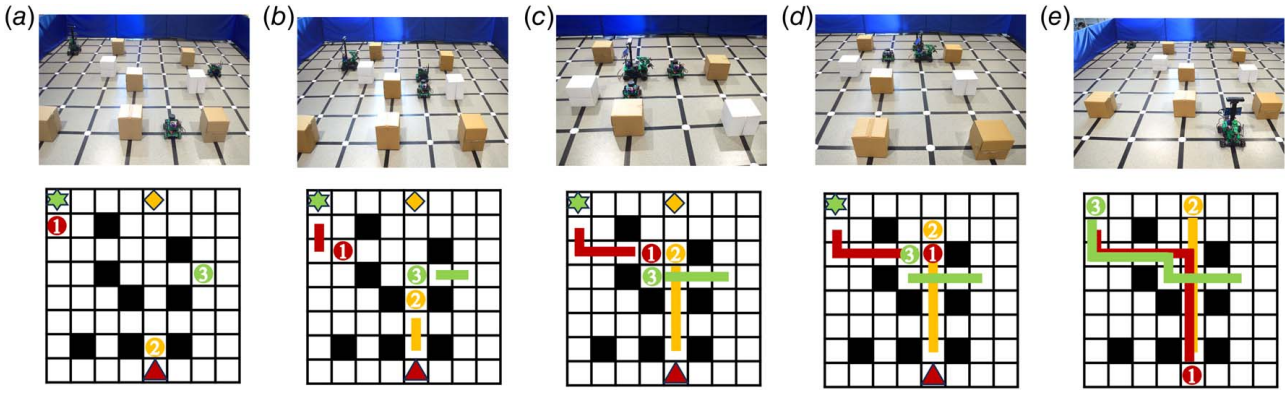

Fig. 10 Snapshots of the robot's path planning in the real-world experiment

5 Conclusion

This study presents a decentralized MAPP approach based on imitation learning and selective communication, aiming to enhance the performance of large-scale mobile robot systems in complex environments. Our extensive simulations in diverse grid environments have shown that our approach consistently outperforms other IL-based methods. It not only achieves an impressive success rate of approximately 97% but also maintains the lowest flowtime variation, consistently below 0.061. Additionally, our approach markedly reduces communication frequency, maintaining a substantial 58% reduction even in environments with extreme robot density. This consistent performance underscores the balance our method strikes between optimality and efficiency, demonstrating its practicality and scalability for real-world applications.

Note that, the performance of our approach, rooted in imitation learning, is inherently linked to the expert algorithms it emulates. The lack of theoretical guarantees poses challenges in achieving a 100% success rate, and the dependence on training data quality and diversity limits its generalization in rare or unrepresented scenarios. Future research can explore the integration of data-driven learning with rule-based systems to provide stronger theoretical guarantees and diversified neural network architectures for more efficient handling of multi-agent interactions and environmental dynamics.

Moreover, the capability to model MAPP problems as Markov decision processes positions RL as an effective alternative to MAPP problems. In RL, an agent can learn to find the path by optimizing its movement actions based on the feedback received in the form of rewards when interacting with the environment. Thus, another future research direction is to investigate the potential of RL in MAPP to relax the dependency of model training on expert-derived knowledge with enhanced algorithmic resilience and flexibility. Central to this effort will be the thoughtful crafting of the reward function to ensure that the resulting behaviors of agents are safe and appropriate. Integrating IL and RL also offers a compelling research pathway, which may lead to a model that not only learns from expert experiences but also can adapt to complex, previously unseen scenarios. Designing an appropriate mechanism for their effective and efficient integration poses a significant challenge for future research.

Acknowledgment

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

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Nomenclature

H = number of shared features
K = number of heads in the multi-head attention mechanism
M = number of agents
T = completion time for agent to reach its goal vertex
c_f = communication frequency
f_v = flowtime variation

l_w = width of the map
l_h = height of the map
n_{random} = number of random cases selected from the training set for rechecking
n_{success} = number of successful cases
n_{total} = total number of tested cases
r_w = width of local field-of-view
r_h = height of local field-of-view
r_d = threshold value for deviation ratio
s_r = success rate
E_p = time taken by expert algorithms for all agents to reach their respective goals
L_p = time taken by learning-based algorithms for all agents to reach their respective goals
δ = visit count threshold
η = predetermined steps an agent takes under expert guidance
κ = epoch interval at which random cases are selected from the training set
ρ_r = robot density on a predefined map
ω_E = optimality bound factor for enhanced conflict-based search

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] Poudel, L., Blair, C., McPherson, J., Sha, Z., and Zhou, W., 2020, "A Heuristic Scaling Strategy for Multi-Robot Cooperative Three-Dimensional Printing," ASME J. Comput. Inf. Sci. Eng., **20**(4), p. 041002.
[3] Ghassemi, P., and Chowdhury, S., 2020, "An Extended Bayesian Optimization Approach to Decentralized Swarm Robotic Search," ASME J. Comput. Inf. Sci. Eng., **20**(5), p. 051003.
[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 (IROS), San Francisco, CA, Sept. 25–30, IEEE, pp. 3260–3267.
[6] Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., et al., 2021, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," Proc. Int. Symp. Comb. Search, **10**(1), pp. 151–158.
[7] 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 Rob. Autom. Lett., **4**(3), pp. 2378–2385.
[8] 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), Las Vegas, NV, Oct. 25–29, pp. 11785–11792.
[9] Barer, M., Sharon, G., Stern, R., and Felner, A., 2021, "Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-agent Pathfinding Problem," Proc. Int. Symp. Comb. Search, **5**(1), pp. 19–27.
[10] van den Berg, J., Guy, S. J., Lin, M., and Manocha, D., 2009, "Reciprocal n-Body Collision Avoidance," The 14th International Symposium of Robotics Research ISRR, Lucerne, Switzerland, Aug. 31–Sept. 3, 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, Auckland, New Zealand, May 9–13, pp. 1804–1808.
[12] Lin, S., Liu, A., Wang, J., and Kong, X., 2022, "A Review of Path-Planning Approaches for Multiple Mobile Robots," Machines, **10**(9), p. 773.
[13] Li, Q., Lin, W., Liu, Z., and Prorok, A., 2021, "Message-Aware Graph Attention Networks for Large-Scale Multi-Robot Path Planning," IEEE Rob. Autom. Lett., **6**(3), pp. 5533–5540.
[14] Kool, W., van Hoof, H., and Welling, M., 2019, "Attention, Learn to Solve Routing Problems!," The 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, May 6–9.
[15] Paul, S., Ghassemi, P., and Chowdhury, S., 2022, "Learning Scalable Policies Over Graphs for Multi-Robot Task Allocation Using Capsule Attention Networks," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, May 23–27, pp. 8815–8822.
[16] Wang, Z., Liu, C., and Gombolay, M., 2022, "Heterogeneous Graph Attention Networks for Scalable Multi-Robot Scheduling With Temporospatial Constraints," Auton. Rob., **46**(1), pp. 249–268.
[17] Paul, S., Li, W., Smyth, B., Chen, Y., Gel, Y., and Chowdhury, S., 2023, "Efficient Planning of Multi-Robot Collective Transport Using Graph Reinforcement Learning With Higher Order Topological Abstraction," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, May 29–June 2, pp. 5779–5785.
[18] 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), Xi'an, China, May 30–June 5, pp. 8699–8705.
[19] 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, Vancouver, BC, Canada, Dec. 6–12, pp. 17271–17282.
[20] 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, Vancouver, BC, Canada, Dec. 6–12, pp. 22069–22079.
[21] Ma, Z., Luo, Y., and Pan, J., 2022, "Learning Selective Communication for Multi-Agent Path Finding," IEEE Rob. Autom. Lett., **7**(2), pp. 1455–1462.
[22] Sheng, J., Wang, X., Jin, B., Yan, J., Li, W., Chang, T.-H., Wang, J., and Zha, H., 2022, "Learning Structured Communication for Multi-agent Reinforcement Learning," Auton. Agent Multi-Agent Syst., **36**(2), p. 50.
[23] 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. Ind. Inform., **19**(10), pp. 10233–10243.
[24] Simonyan, K., and Zisserman, A., 2015, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, May 7–9.
[25] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., 2014, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," NIPS 2014 Workshop on Deep Learning, Montreal, Quebec, Canada, Dec. 8–13.
[26] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y., 2018, "Graph Attention Networks," The 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, Apr. 30–May 3.
[27] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I., 2017, "Attention Is All You Need," Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, Dec. 4–9, pp. 6000–6010.
[28] Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A., 2018, "End-to-End Driving Via Conditional Imitation Learning," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 21–25, pp. 4693–4700.