LEARNING TO COMMUNICATE LOCALLY FOR LARGE-SCALE MULTI-AGENT PATHFINDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent pathfinding (MAPF) is a widely used abstraction for multi-robot trajectory planning problems, where multiple homogeneous agents move simultaneously within a shared environment. Although solving MAPF optimally is NP-hard, scalable and efficient solvers are critical for real-world applications such as logistics and search-and-rescue. To this end, the research community has proposed various decentralized suboptimal MAPF solvers that leverage machine learning. Such methods frame MAPF (from a single agent perspective) as a Dec-POMDP where at each time step an agent has to decide an action based on the local observation and typically solve the problem via reinforcement learning or imitation learning. We follow the same approach but additionally introduce a learnable communication module tailored to increase the level of cooperation between the agents via efficient feature sharing. We present the Local Communication for Multi-agent Pathfinding (LC-MAPF), a generalizable pre-trained model that applies multi-round communication between neighboring agents to exchange information and improve their coordination. Our experiments show that the introduced method outperforms the existing learning-based MAPF solvers, including IL and RL-based approaches, across diverse metrics in a diverse range of (unseen) test scenarios. Remarkably, the introduced communication mechanism does not compromise LC-MAPF's scalability, a common bottleneck for communication-based MAPF solvers.

1 INTRODUCTION

Modern robotic systems often involve multiple mobile agents navigating shared environments, such as warehouse robots (Li et al., 2021a) or autonomous vehicles (Li et al., 2023). A key abstraction for coordinating such agents is multi-agent pathfinding (MAPF) (Stern et al., 2019).

In MAPF, time is discrete and agents move on a graph (typically a 4-connected grid). Each agent synchronously either moves to a neighboring vertex or waits, taking one time step per action. The goal is to compute collision-free individual plans for all agents.

Many real-world challenges – continuous space and time, asynchronous behavior, limited communication and observation – are not directly captured by MAPF. Despite these simplifications, MAPF successfully models the central difficulty: coordinating agents to avoid collisions while optimizing a cost function. Several studies demonstrated successful application of MAPF-based methods to continuous, noisy real-world environments (Hönig et al., 2016; Ma et al., 2019a).

MAPF is most commonly solved in a centralized setting, where a single planner with full knowledge generates plans for all agents. Many optimal and suboptimal centralized solvers have been proposed (Sharon et al., 2015; Surynek et al., 2016; Okumura et al., 2022; Okumura, 2023; Li et al., 2022; Veerapaneni et al., 2024b; Wang et al., 2025). Optimal solvers scale poorly as MAPF is NP-hard (Surynek, 2010), while suboptimal solvers scale to thousands of agents but with degraded solution quality. Balancing computational efficiency and solution quality remains a central research focus.

A promising alternative is decentralized approach, whereagents independently select actions based on local observations. The decision-making policy may be fully learned or hybrid (Li et al., 2021b;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

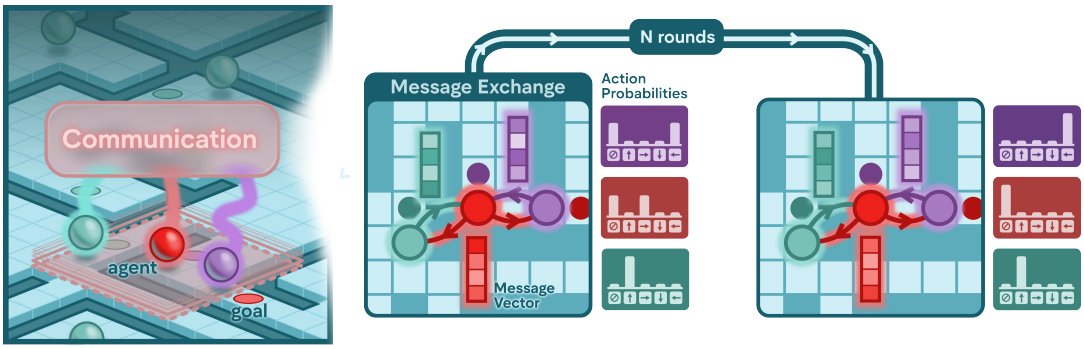


Figure 1: In the proposed LC-MAPF, agents iteratively exchange messages to progressively refine their predicted action distributions over multiple rounds.

Wang et al., 2023; Ma et al., 2021a;b; Skrynnik et al., 2024; 2023; Phan et al., 2025; Jain et al., 2025; 2026). A recent survey provides a comprehensive overview (Alkazzi & Okumura, 2024).

A notable advancement is MAPF-GPT (Andreychuk et al., 2025b), which relies entirely on supervised learning using a transformer trained on approximately one billion observation-action pairs. Despite its simplicity, MAPF-GPT outperforms numerous state-of-the-art learning-based methods. However, it lacks agent-to-agent communication – while it learns collaborative behavior through data, agents cannot explicitly coordinate during problem-solving.

Several decentralized methods (MAGAT (Li et al., 2021b), SCRIMP (Wang et al., 2023), DHC (Ma et al., 2021a), DCC (Ma et al., 2021b)) use communication, but mostly limited to sharing local observations in a single round (Alkazzi & Okumura, 2024), falling short of enabling meaningful coordination (with few exceptions like in DHC).

We argue that effective communication in decentralized MAPF should involve multi-round interaction, enabling agents to improve coordination and reduce conflicts rather than relying on single-round information sharing. Motivated by this, we explore how to equip a large transformer-based imitation learning model with effective local communication.

Our main contributions are:

- We introduce LC-MAPF, a decentralized, imitation-learned MAPF method whose communication keeps the message channel separate from the observation and re-injects the observation into the decoder each round. This decoupled design makes multi-round local communication cheap to run, letting LC-MAPF combine it with scalability to thousands of agents.
- We present a transformer-based model with 3 million parameters that significantly improves performance and sets a new state-of-the-art among learnable decentralized MAPF solvers, with extensive evaluations against existing learning-based approaches.
- We empirically validate LC-MAPF's communication design, showing that four local communication rounds yield the best, tested performance and that the method preserves linear inference-time scaling up to 5,000 agents.

2 Related Work

2.1 Foundation Models for Multi-Agent Systems

Foundation models are typically trained on large-scale datasets, enabling generalization through zero-shot or few-shot learning. For autonomous agents, demonstrations of task execution in the environment are used as a dataset, and generalization implies the execution of new tasks that were not in the training data distribution without additional demonstrations or with a minimal number of them (Firoozi et al., 2023). Demonstration-based pretraining is not commonly used in multi-agent settings, but there are some examples, including games such as chess (Silver et al., 2016),

cooperative video games via self-play (Berner et al., 2019), and multi-agent pathfinding, as in SCRIMP (Wang et al., 2023).

A key strength of foundation models is their fine-tuning capability, which supports rapid adaptation to task-specific requirements. While widely adopted in robotics, particularly in multimodal tasks involving text-based instructions (Firoozi et al., 2023), their use in multi-agent systems remains relatively limited. Notable examples include the Magnetic-One model for language and multimodal tasks in WebArena (Fourney et al., 2024) and MAPF-GPT for decentralized pathfinding (Andreychuk et al., 2025b).

2.2 Multi-agent Pathfinding

A variety of approaches have been proposed for solving MAPF. Rule-based solvers are designed for fast computation but lack guarantees on solution quality (Okumura, 2023; Li et al., 2022). Reduction-based methods convert MAPF into classical problems such as minimum-cost flow or SAT, leveraging existing solvers to compute optimal solutions (Surynek et al., 2016). Search-based solvers, such as CBS and its variants (Sharon et al., 2015), apply graph search techniques and often offer optimality or bounded-suboptimality guarantees. Simpler methods like prioritized planning (Ma et al., 2019b) trade off optimality for efficiency and scalability.

2.3 Communication-based MAPF Methods

More recently, learning-based approaches have emerged. PRIMAL (Sartoretti et al., 2019) was among the first to demonstrate decentralized MAPF solving via learning. In the case of PRIMAL the only communication between agents is their corresponding targets. One of the first learnable MAPF solvers that has a specific learnable communication block was DHC (Ma et al., 2021a) that demonstrated significant improvement over PRIMAL. Subsequent methods such as DCC (Ma et al., 2021b) utilize the ideas proposed by DHC, but enhance the communication mechanism by training selective communication. Another approach, SCRIMP (Wang et al., 2023), combines imitation learning, reinforcement learning and communication mechanism and improves the efficiency even further.

Another line of communication-based methods builds on graph attention networks. MAGAT (Li et al., 2021b) replaces fixed communication structures with a graph attention mechanism, allowing agents to dynamically weight messages from neighbors for stronger coordination. Recently, multiple improved variants have appeared: HMAGAT (Jain et al., 2026), which utilizes hypergraphs, and MAGAT+ (Jain et al., 2025), an enhanced version of MAGAT that uses three stacked graph attention layers compared to the single layer in the original.

Method	Comm. rounds	Imitation-only training	Variable rounds	Obs. re-injected
DHC (Ma et al., 2021a)	2	✗	✓	✗
MAGAT (Li et al., 2021b)	1	✓	✗	✗
SCRIMP (Wang et al., 2023)	1	✗	✗	✗
MAGAT+ (Jain et al., 2025)	3	✓	✗	✗
HMAGAT (Jain et al., 2026)	3	✗†	✗	✗
LC-MAPF (ours)	**4**	✓	✓	✓

Table 1: Communication-design comparison of LC-MAPF and learnable MAPF baselines (✓/✗: property present/absent). †HMAGAT's policy is imitation-learned; a separate PPO-trained actor–critic module predicts agent-specific softmax temperatures.

Table 1 contrasts LC-MAPF with representative learnable communication methods. End-to-end learned (unsupervised), multi-round, and attention-based communication are each individually not unique to our approach; what is specific to LC-MAPF is the combination of these methods with strictly local communication, the separation of message channel from the observation representation, and the explicit re-injection of observation at every round. Prior methods tie an agent's outgoing message to the same representation it uses for its observation, causing the observation signal to be progressively overwritten as it blends into the evolving message state across rounds; LC-MAPF avoids this by maintaining a separate message vector and re-injecting the observation at every round; we detail the mechanism in Section 4.

3 BACKGROUND

3.1 PROBLEM STATEMENT

MAPF problem is a tuple $(G, s^1, ..., s^n, g^1, ..., g^n)$, where $G = (V, E)$ is a graph representing the environment, $s^i \in V$ is the start vertex of the i-th agent, and $g^i \in V$ is its goal vertex. A total of n agents ($U = \{u_1, \dots, u_n\}$) are present in the environment. The task is to find a set of plans $Pl = \{pl^i\}$ composed of move to an adjacent vertex or stay at the current vertex actions. The plans should be conflict-free, i.e., no two agents occupy the same vertex or traverse the same edge at the same time. The solution cost is typically measured by the ***Sum-of-Costs***, $SOC(Pl) = \sum_{i=1}^{n} cost(pl^i)$, or the ***makespan***, $msn(Pl) = \max_{i=1}^{n} cost(pl^i)$, where $cost(pl^i)$ is the timestep at which agent i reaches its goal.

We consider MAPF as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2002) defined as $M = \langle S, A, U, P, R, O, \mathcal{O}, \gamma \rangle$. Communication graph is a directed asymmetric dynamic graph $G_c = (U, E_c)$, where each agent $u \in U$ communicates with k closest neighbors within its observation range ($k <$ maximal neighbors in observation), leading to the asymmetrical communication. Agents' observability is symmetrical. Each agent's $u \in U$ local observation $o^{(u)} \in O$ is based on the global observation function $\mathcal{O}(s, a) : S \times A \to O$. Given the environment graph is a 4-connected grid, the observation is a $(2r + 1) \times (2r + 1)$ grid patch of radius r centered at the agent's current position. The observation stores local information about environment map and about agent and its neighbors, such as their current positions, goal locations, action history, and next greedy actions.

The communication protocol for each time step t is the following. A message $m \in \mathbb{R}^{d_{latent}}$ is a vector in a latent vector space. Agents sequentially perform R^c communication rounds. At each round r, each agent receives messages from its neighbors and processes them with latent representation of its observation to produce the next round message. During a communication round, all agents produce their messages simultaneously. Then messages are broadcast with respect to the communication graph, and the next round is performed. Communication stops after R^c rounds. Overall, at each time step, each agent $u \in U$ performs R^c communication rounds and after that chooses an action $a^{(u)} \in A$ to form a joint action $\mathbf{j} \in \mathbf{J} = A^n$. This joint action leads to a stochastic change of the environment according to the transition function $P(s'|s, \mathbf{j}) : S \times \mathbf{J} \times S \to [0, 1]$. Next, each agent receives an individual scalar reward $R(s, u, \mathbf{j}) : S \times U \times \mathbf{J} \to \mathbb{R}$, that depends on current state and joint action and may vary for different agents. The discount factor $0 \leq \gamma \leq 0$ determines the importance of future rewards. To make decisions, each agent maintains an action-observation history $\tau^{(u)} \in T = (O \times A)^*$ to condition a stochastic policy $\pi^{(u)}(a^{(u)}|\tau^{(u)}) : T \times A \to [0, 1]$. The task is to learn a homogeneous individual policy $\pi^{(u)}$ shared by all agents that maximizes the expected cumulative reward over time.

The detailed description of the observation information encoding, message vectors initialization, encoding, and update mechanism is presented in Section 4.

3.2 IMITATION LEARNING FOR MAPF

Imitation learning seeks to approximate an expert policy $\hat{\pi}$ by training a parameterized policy π_θ. A dataset $\mathcal{T}$ of expert trajectories is first collected: $\hat{\mathcal{T}} = \{\hat{\tau}_i\}_{i=1}^{K}$, where each $\hat{\tau}_i = \{(s^1, \mathbf{a}^1), \dots, (s^L, \mathbf{a}^L)\}$ consists of state and joint action pairs. In MAPF, $\hat{\pi}$ is typically a centralized solver, for example, LaCAM* (Okumura, 2024).

To enable decentralized learning, individual agent trajectories $\tau_u^{\hat{\pi}} = \{(o_u^1, a_u^1), \dots, (o_u^L, a_u^L)\}$ are extracted, where o_u^t is the local observation of agent u at time t, and a_u^t is the corresponding expert action. Observations may be represented as tensors or token sequences. The resulting dataset $\mathcal{D} = \{\tau_u^{\hat{\pi}}\}_{u=1}^{n}$ is then used to train the policy.

The learning objective minimizes the negative log-likelihood of expert actions:

$$\theta^\star = \arg\min_\theta \mathbb{E}_{(o_u, a_u^{\hat{\pi}}) \sim \mathcal{D}} \left[-\log \pi_\theta(a_u^{\hat{\pi}} \mid o_u) \right]. \tag{1}$$

After training, actions are sampled as $a^u \sim \pi_\theta(o_u)$.

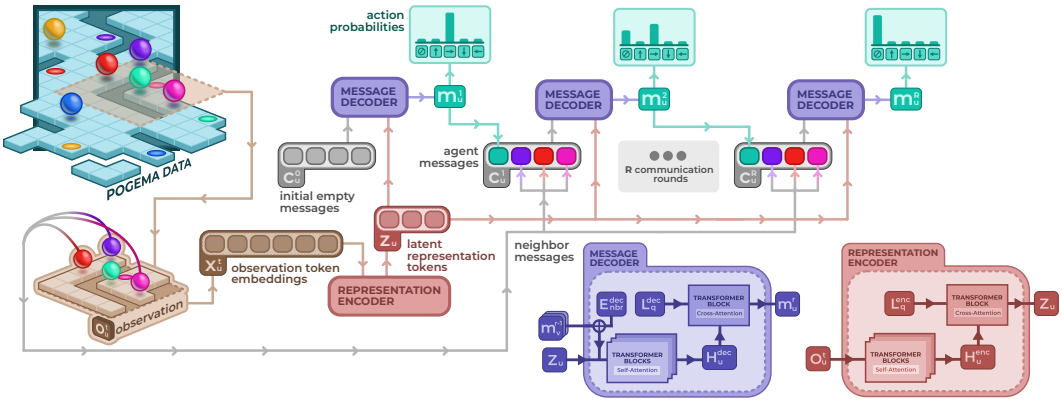


Figure 2: Overview of the LC-MAPF architecture. Agents encode observations into a compact latent state z_u. This representation is iteratively refined over R communication rounds via a Transformer-based decoder that cross-attends to neighbor messages m_v^r, yielding the final action probability distribution p_u.

4 METHOD

The overall communication and action prediction workflow is illustrated in Fig. 2. At each time step $t \in [1, \dots, L]$ and for each agent $u \in [1, \dots, U]$, the model receives observation

$$o_u^t = [\text{cost-to-go}_u^t,\ i_u^t,\ n_{u,1}^t, \dots, n_{u,k}^t], \tag{2}$$

where cost-to-go_u^t is an egocentric cost-to-go matrix, i_u^t contains the agent's own features (relative positions of current and goal locations, greedy action, and previous k actions), and each $n_{u,j}^t$ contains analogous information for one of the k closest neighboring agents. This localized, tokenized representation allows each agent to reason using only information that is relevant for preventing collisions and coordinating movement. A tokenized observation has up to 256 tokens: first 121 tokens store the 11×11 local field of view of radius 5. The next 130 tokens store the information about the features of the agent and up to its 12 neighbors (13 agents overall, 10 tokens per agent): 2 tokens encode current position, 2 – goal location, 5 – action history, and 1 – next greedy action. The last 5 tokens are padding tokens. Agent can communicate with all neighbors present in its observation.

The observation is converted into an embedding sequence:

$$X_{0,u} = E_{\text{tok}}(o_u^t) + E_{\text{pos}} + E_{\text{nbr}}, \tag{3}$$

where E_{tok} is a learnable embedding matrix, E_{pos} is an integer vector of token positions in the input sequence, and E_{nbr} serves as a neighbor identifier so that the model can distinguish which nearby agent contributed each token.

A Transformer encoder processes this embedded input and produces contextualized representations:

$$H_u^{\text{enc}} = \text{Encoder}(X_{0,u}). \tag{4}$$

To avoid propagating the entire observation sequence throughout communication, we apply an information bottleneck inspired by the Perceiver architecture (Jaegle et al., 2022). A small set of learnable latent queries L_q^{enc} cross-attends to encoded tokens and produces compact latent state:

$$z_u = \text{LatentEncoder}(L_q^{enc}, H_u^{\text{enc}}), \qquad z_u \in \mathbb{R}^{T_{\text{latent}} \times d_{\text{latent}}}. \tag{5}$$

This forms the agent's internal representation of the world and only this compressed state participates in communication, making the communication cost independent of the observation size. Here T_{latent} denotes the number of latent tokens, and d_{latent} is the latent dimension. The agents then perform R_{comm} rounds of local communication. Each agent stores a message vector $m_u^r \in \mathbb{R}^{d_{\text{latent}}}$ for round r, initialized with a learnable vector m_u^0 shared across all agents.

At round r, agent u receives a message m_v^{r-1} from each neighbor $v \in \mathcal{N}(u)$. These messages are concatenated into a sequence, and each message is tagged with its sender's embedding $E_{\text{nbr}}^{dec}(v)$ to

track which agent it came from.

$$C_u^r = \{\, m_v^{r-1} + E_{\text{nbr}}^{dec}(v) \,\}_{v \in \mathcal{N}(u) \cup \{u\}}. \tag{6}$$

Messages are inserted into a decoding module together with the agent's latent state:

$$h_u^r = \text{Decoder}(L_q^{dec}, [z_u, C_u^r]). \tag{7}$$

The decoder integrates information from neighbors and produces a new message:

$$m_u^r = \text{MsgHead}(h_u^r). \tag{8}$$

After final round, prediction head produces action logits:

$$a_u = \text{ActionHead}(h_u^{R_{\text{comm}}}), \qquad p_u = \text{softmax}(a_u). \tag{9}$$

Our Transformer blocks utilize the Differential Transformer architecture (Ye et al., 2025), which integrates a differential attention mechanism, RMSNorm, and SwiGLU, further enhanced by the HybridNorm normalization scheme (Zhuo et al., 2026).

LC-MAPF is trained from expert demonstrations in a fully end-to-end manner with the cross-entropy objective:

$$\mathcal{L} = \text{CE}\,(a_u, a_{u_*}) \tag{10}$$

where a_{u_*} is the one-hot expert action. The total loss is averaged across all agents in the batch.

A key property of LC-MAPF is that messages are not supervised. There is no auxiliary loss on m_u^r, nor are the messages forced to represent explicit semantic content. Instead, messages influence future rounds of communication, and therefore their gradients flow through the action loss of the agents that receive them. During backpropagation, the update to m_u^r depends on how it affects the action logits of neighboring agents in subsequent rounds. Consequently, the communication signal is shaped implicitly by the action loss of the agents that receive the messages, without requiring explicit supervision. Separating the observation latent z_u from the message m_u^r and re-injecting z_u at every round let each agent refine a dedicated message while retaining access to its own observation throughout communication. Because communication operates on the fixed-size latent z_u rather than the full observation sequence, the cost of each additional round is small, so several rounds remain affordable; this keeps LC-MAPF tractable even with thousands of agents. In all our experiments we use $R_{\text{comm}} = 4$ communication rounds.

5 Experimental Results

To evaluate LC-MAPF empirically, we conducted multiple series of experiments. The main series aimed at face-to-face comparison of LC-MAPF to state of the art in learnable MAPF, specifically to MAPF-GPT (Andreychuk et al., 2025b), its recent fine-tuned variation MAPF-GPT-DDG (Andreychuk et al., 2025a), SCRIMP (Wang et al., 2023), DCC (Ma et al., 2021b), HMAGAT (Jain et al., 2026), and MAGAT+ (Jain et al., 2025). The experiments were conducted on the POGEMA benchmark (Skrynnik et al., 2025) (Random, Mazes, Warehouse, and Cities Tiles maps) following the original MAPF-GPT evaluation protocol. LC-MAPF was trained on Random, Mazes, and Houses maps' instances (a 0.6:0.2:0.2 split) with up to 32 agents. All generators for the corresponding map types are included in the code repository. It should be noted that Houses maps' generator was taken from (He et al., 2024). Expert demonstrations were generated using LaCAM* with a time budget of 10 seconds per instance. Regarding evaluation maps, Random and Mazes match training types, Warehouse and Cities Tiles differ significantly in topology and are used to evaluate out-of-distribution generalization; agent counts beyond 32 are likewise unseen during training. Mazes and Random maps range in size from 17×17 to 21×21, and contain up to 80 and 96 agents respectively. The Warehouse type (maximum 192 agents) features a single map of size 33×46 with restrictions on agent start and goal locations to model real-world fulfillment scenarios. The Cities Tiles maps are of 64×64 size, allowing for up to 256 agents. In all runs the episode length was limited to 128 steps, except for Cities Tiles, where the episode length was 256. More details about the benchmark and evaluation protocol can be found in (Skrynnik et al., 2025). We do not employ collision shielding (Veerapaneni et al., 2024a) in evaluation, neither for LC-MAPF nor for the baselines, as such post-processing is centralized and modifies agents' actions, making it difficult to assess the true performance of the learned policy in our decentralized setting.

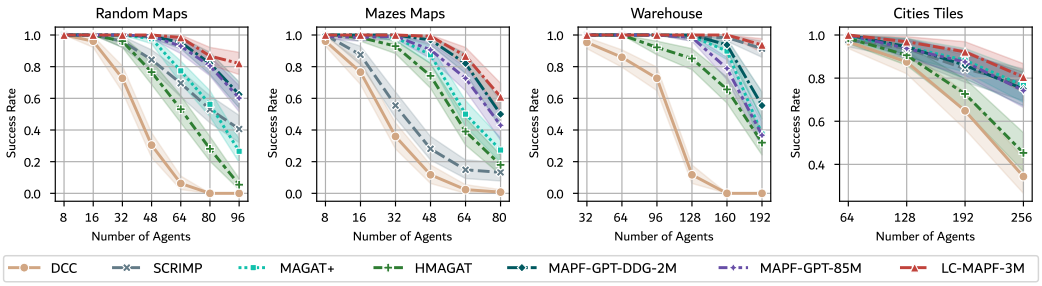


Figure 3: Success rates of the approaches on different map types depending on the number of agents in the instances (higher is better). The shaded area indicates the 95% confidence interval.

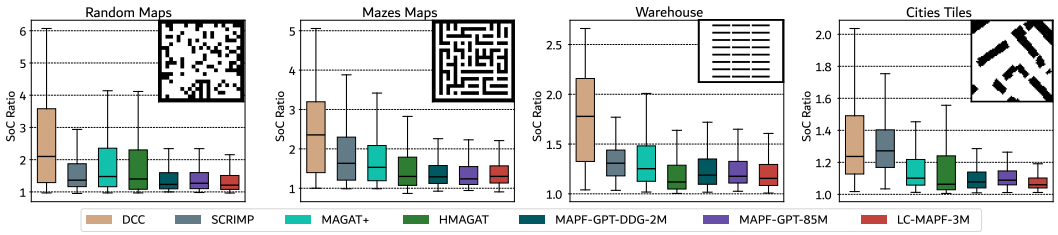


Figure 4: SoC ratio relative to solutions found by the LaCAM* approach (lower is better).

5.1 Comparison with the Baselines

Fig. 3 shows that LC-MAPF is on par or better than baselines in terms of the average success rate, including original 85M MAPF-GPT model and fine-tuned MAPF-GPT-DDG. Fig. 4 presents the ratio of SoC (solution cost) relative to the centralized LaCAM* solution. LC-MAPF achieves the best results in most of the cases. Surprisingly, while original MAPF-GPT has lower success rate than LC-MAPF on Mazes maps, its relative median solution cost is slightly better – 1.24 vs 1.3. Although, the average relative solution cost for MAPF-GPT-85M on Mazes maps is 1.42, compared to 1.4 for LC-MAPF. On Warehouse map HMAGAT is able to solve almost as many instances as LC-MAPF and its solutions even have slightly better relative SoC ratio, but on the rest of the maps it substantially underperforms.

5.2 Communication Rounds Ablation Study

To estimate the influence of communication mechanism on performance, we tested LC-MAPF with 1-8 communication rounds on Warehouse map with populations of 32-192 agents. The results are shown in Table 2. LC-MAPF needs at least 2 communication rounds to solve any instance. The best performance is achieved with 4 communication rounds matching the communication configuration during training. Increasing the number of rounds during inference helps to reduce inter-agent collisions.

	Success rate across different LC-MAPF communication rounds				
Agents	$R=1$	$R=2$	$R=4$	$R=6$	$R=8$
32	0.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
64	0.00 ±0.00	0.77 ±0.07	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
96	0.00 ±0.00	0.09 ±0.05	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
128	0.00 ±0.00	0.01 ±0.01	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
160	0.00 ±0.00	0.02 ±0.02	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
192	0.00 ±0.00	0.00 ±0.00	0.94 ±0.04	0.88 ±0.05	0.94 ±0.04

	Collision counts across different LC-MAPF communication rounds				
Agents	$R=1$	$R=2$	$R=4$	$R=6$	$R=8$
32	299 ±6	62 ±3	5 ±1	8 ±1	7 ±1
64	788 ±7	258 ±7	25 ±2	42 ±2	30 ±2
96	1362 ±14	618 ±16	79 ±4	120 ±5	93 ±5
128	2078 ±26	1145 ±46	227 ±14	292 ±13	253 ±13
160	3317 ±90	2108 ±82	487 ±24	639 ±31	570 ±32
192	5330 ±143	3557 ±137	1175 ±77	1409 ±80	1225 ±68

Table 2: Success Rate and Number of Collisions of Different Versions of LC-MAPF on Warehouse Map. The provided values are average±95% confidence interval. Best mean values are marked with background color per row.

5.3 Message Failure Test

Communication in Dec-POMDPs has been studied under realistic constraints such as delays, failures, and costs (Lauri et al., 2019). Here, we evaluate LC-MAPF under message transmission failures. For each agent and communication round, the updated message is replaced, with a given probability, by a random vector sampled from the standard normal distribution. We test LC-MAPF with 20% and 50% message failure on Random maps from the POGEMA benchmark and compare it with the original model, with results in Table 3.

	0% failure		20% failure		50% failure	
Ag.	Succ.	Coll.	Succ.	Coll.	Succ.	Coll.
24	1.0 ±0.0	7 ±2	1.0 ±0.0	13 ±2	1.0 ±0.0	30 ±5
32	1.0 ±0.0	18 ±5	1.0 ±0.0	31 ±5	1.0 ±0.0	81 ±18
48	1.0 ±0.0	72 ±14	1.0 ±0.0	138 ±34	0.9 ±0.0	300 ±61
64	1.0 ±0.0	219 ±47	0.9 ±0.0	431 ±88	0.8 ±0.1	820 ±139

Table 3: Communication Failure Results on `Random` Maps. Zero message failure means original LC-MAPF performance. Provided values are average ± 95% confidence interval. Best results are marked with background color per row.

The success rate comparison reveals the negative impact of random noise messages compared to the LC-MAPF for 48 agents and above, highlighting the importance of the information conveyed in LC-MAPF messages for achieving performance improvement. However, despite the extreme setup with 50% message failure probability, agents can successfully solve simpler tasks with up to 32 agents, and obtain partial success in cases with larger agent populations.

5.4 Communication Bandwidth

LC-MAPF enables communication between an agent and a fixed set of 12 local neighbors, chosen because collisions are only possible with agents occupying the 12 closest cells. Agents in these cells are prioritized, though communication is not strictly restricted to them. Fig. 5 shows locations and actions that may cause collisions. The first four agents are in cardinally adjacent cells (via swaps or waits), the next four in diagonal cells (if both enter the same cell), and the last four are two cells away (depending on actions). Agents in any other location cannot collide in the current step.

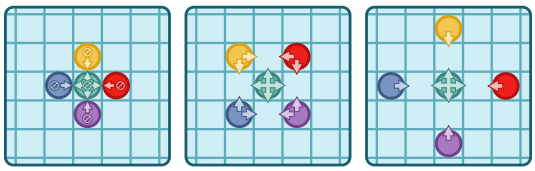


Figure 5: Agents and corresponding actions that may collide with the green reference agent.

We acknowledge that limited communication can affect performance, but this constraint reflects practical considerations: real-world communication bandwidth is limited and incurs costs (e.g., battery drain), creating an inherent trade-off between performance and bandwidth. This limitation can be partially mitigated through chain communication: agent A communicates with agent B at round t, and B subsequently communicates with agent C at round $t+1$, propagating information from A to C with some delay. To validate the neighborhood size, we test more restrictive limits on Mazes maps from the POGEMA benchmark, limiting communicating agents to 1, 2, 4, 8, and 13 sorted by distance. For example, Limit $= 2$ means each agent receives messages from at most the 2 closest agents (including itself). Results, presented in Table 4, show that limiting communicating agents significantly reduces success rate, especially with strict limitations (4 or fewer) on larger populations (48 and 64 agents). Tighter limits have minimal impact on smaller instances because the actual number of nearby agents typically satisfies the reduced limit. This highlights the importance of communication for LC-MAPF.

Agents	$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 13$
24	0.41 ±0.08	0.91 ±0.05	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
32	0.28 ±0.08	0.74 ±0.07	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
48	0.10 ±0.05	0.38 ±0.08	0.86 ±0.06	0.96 ±0.04	0.98 ±0.03
64	0.06 ±0.04	0.18 ±0.07	0.58 ±0.09	0.79 ±0.07	0.87 ±0.06

Table 4: Success Rates for Different Communication Neighborhood Sizes on Mazes Maps. The Limit value shows number of communicating agents. Limit = 13 is original LC-MAPF. Values are average ± 95% confidence interval. Best results are marked with background color per row.

5.5 LARGE-SCALE EVALUATION

The main experiments were conducted on the POGEMA benchmark, with instances containing up to 256 agents. To demonstrate that LC-MAPF can scale to significantly larger instances, we evaluated it on 256×256 random maps (obstacle densities of 10%, 20%, 30%) and empty maps (0%) with up to 5,000 agents. Results are shown in Table 5. Reported numbers correspond to the makespan, i.e., the number of steps for all agents to reach their goals simultaneously. A value of 2048 indicates the instance was not solved and was terminated. LC-MAPF scales linearly with the number of agents, requiring approximately 50 ms per step per each 1,000 agents regardless of the density of the obstacles on the map. All inference was performed in a single batched forward pass on a GPU.

	Density				
Agents	0%	10%	20%	30%	Run. (ms)
1000	474	476	486	532	49 ±1
2000	456	497	495	2048	94 ±3
3000	476	489	521	2048	145 ±4
4000	487	488	531	2048	200 ±5
5000	508	500	559	2048	255 ±5

Table 5: Large-scale evaluation of LC-MAPF.

5.6 ROBOTICS EXPERIMENTS

As an experimental validation, we additionally deploy LC-MAPF on a real multi-robot platform, reproducing a POGEMA maze benchmark with three physical robots running the policy over actual ROS 2 communication. LC-MAPF controls the robots without modification and reproduces effective real-world coordination and goal achievement; full platform, integration, and protocol details are given in Appendix C.

6 CONCLUSION

We introduced LC-MAPF, a transformer-based framework with communication learning for decentralized multi-agent pathfinding. Its communication runs in multiple local rounds, trained end-to-end from expert demonstrations without explicit message supervision. Keeping the inter-agent message channel separate from the observation representation and routing it through a fixed-size bottleneck makes multi-round communication inexpensive as the number of agents grows. On the POGEMA benchmark, LC-MAPF matches or outperforms state-of-the-art learning-based MAPF solvers across diverse scenarios.

LC-MAPF maintains linear scalability with the number of agents, overcoming a common limitation of communication-based approaches. Ablation studies confirm that multi-round local communication enhances performance without sacrificing scalability or generalization. These results highlight LC-MAPF as a generalizable pre-trained model that offers an effective and scalable solution for decentralized multi-agent pathfinding through multi-round local communication.

AI USE STATEMENT

In this work, we have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving mathematical claims or to assist in the writing of proofs, to propose or refine hypotheses, to design or provide feedback on our research methodology or experiments, or to interpret results; translation, dataset cleaning and reformatting, and qualitative and thematic data analysis are not applicable to this work. We used generative AI tools for editing software code, for drafting and editing parts of this paper to improve its readability and formatting, and for identifying and summarizing relevant literature. We have reviewed all AI-assisted work. The research idea, the LC-MAPF communication design, and the architecture of each component were designed and specified by the authors; all AI-assisted code was reviewed and tested by the authors, and every experimental number reported in this paper was produced by running the code; all AI-assisted text was read and revised by the authors; and all references were added manually by the authors, who checked each one against its original source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

The LC-MAPF architecture, communication mechanism, and training procedure are specified in Section 4. Training infrastructure, optimizer settings, and the complete hyperparameter configuration are reported in Appendix D. The message-failure evaluation protocol is described in Appendix A, and the comparison against reactive search-based methods, including solver budgets and configurations, in Appendix B. For the physical experiments, the robot platform, software stack, discrete-to-continuous execution bridge, and experimental protocol are documented in Appendix C.

An anonymized code and data supplement accompanies this submission. It provides the LC-MAPF implementation; the trained LC-MAPF-3M checkpoint; evaluation configurations and map sets; the two-phase dataset-generation pipeline that produces expert trajectories and observation–action pairs; and the training script and configuration. A Dockerfile and a pinned requirements file are included so the environment can be reconstructed, and the accompanying README gives the commands for inference, benchmark evaluation, dataset generation, and training. The supplement also contains raw per-instance benchmark logs, so the reported comparisons can be recomputed without rerunning the experiments.

References

Jean-Marc Alkazzi and Keisuke Okumura. A comprehensive review on leveraging machine learning for multi-agent path finding. *IEEE Access*, 2024.

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. Advancing learnable multi-agent pathfinding solvers with active fine-tuning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10564–10571, 2025a.

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. MAPF-GPT: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23126–23134, 2025b.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Dkebiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002. doi: 10.1287/moor.27.4.819.297.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 2023.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Chengyang He, Tianze Yang, Tanishq Duhan, Yutong Wang, and Guillaume Sartoretti. Alpha: Attention-based long-horizon pathfinding in highly-structured areas. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14576–14582. IEEE, 2024.

Wolfgang Hönig, TK Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of The 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pp. 477–485, 2016.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

Rishabh Jain, Keisuke Okumura, Michael Amir, and Amanda Prorok. Graph attention-guided search for dense multi-agent pathfinding. *arXiv preprint arXiv:2510.17382*, 2025.

Rishabh Jain, Keisuke Okumura, Michael Amir, Pietro Lio, and Amanda Prorok. Pairwise is not enough: Hypergraph neural networks for multi-agent pathfinding. In *The Fourteenth International Conference on Learning Representations*, 2026.

Mikko Lauri, Joni Pajarinen, and Jan Peters. Information gathering in decentralized pomdps by policy graph improvement. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1143–1151, 2019.

Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, pp. 11272–11281, 2021a.

Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J Stuckey, and Sven Koenig. Mapf-lns2: Fast repairing for multi-agent path finding via large neighborhood search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10256–10265, 2022.

Jiaoyang Li, Eugene Lin, Hai L Vu, Sven Koenig, et al. Intersection coordination with priority-based search for autonomous vehicles. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, pp. 11578–11585, 2023.

Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021b.

Runzhe Liang, Rishi Veerapaneni, Daniel Harabor, Jiaoyang Li, and Maxim Likhachev. Real-time lacam for real-time mapf. In *Proceedings of the Eighteenth International Symposium on Combinatorial Search*, volume 18, pp. 196–200, 2025. doi: 10.1609/socs.v18i1.35993.

H. Ma, W. Hönig, T. K. S. Kumar, N. Ayanian, and S. Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pp. 7651–7658, 2019a.

Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7643–7650, 2019b.

Ziyuan Ma, Yudong Luo, and Hang Ma. Distributed heuristic multi-agent path finding with communication. In *2021 IEEE International Conference on Robotics and Automation (ICRA 2021)*, pp. 8699–8705, 2021a.

Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters*, 7(2):1455–1462, 2021b.

Keisuke Okumura. Lacam: Search-based algorithm for quick multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11655–11662, 2023.

Keisuke Okumura. Engineering lacam*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1501–1509, 2024.

Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022.

Thomy Phan, Timy Phan, and Sven Koenig. Generative curricula for multi-agent path finding via unsupervised and reinforcement learning. *Journal of Artificial Intelligence Research*, 82:2471–2534, 2025.

Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219:40–66, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr I Panov. When to switch: planning and learning for partially observable multi-agent pathfinding. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Alexey Skrynnik, Anton Andreychuk, Maria Nesterova, Konstantin Yakovlev, and Aleksandr Panov. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*, 2024.

Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin Yakovlev, and Aleksandr Panov. POGEMA: A benchmark platform for cooperative multi-agent pathfinding. In *The Thirteenth International Conference on Learning Representations*, 2025.

Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS 2019)*, pp. 151–158, 2019.

Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, pp. 1261–1263, 2010.

Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Efficient sat approach to multi-agent path finding under the sum of costs objective. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pp. 810–818. IOS Press, 2016.

Rishi Veerapaneni, Arthur Jakobsson, Kevin Ren, Samuel Kim, Jiaoyang Li, and Maxim Likhachev. Work smarter not harder: Simple imitation learning with cs-pibt outperforms large scale imitation learning for mapf. *arXiv preprint arXiv:2409.14491*, 2024a.

Rishi Veerapaneni, Qian Wang, Kevin Ren, Arthur Jakobsson, Jiaoyang Li, and Maxim Likhachev. Improving learnt local mapf policies with heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pp. 597–606, 2024b.

Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9301–9308, 2023.

Yutong Wang, Tanishq Duhan, Jiaoyang Li, and Guillaume Sartoretti. Lns2+ rl: Combining multi-agent reinforcement learning with large neighborhood search in multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23343–23350, 2025.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. In *International Conference on Learning Representations*, volume 2025, pp. 144–164, 2025.

Zhijian Zhuo, Yutao Zeng, Ya Wang, Sijun Zhang, Xiaoqing Li, Jian Yang, Jinwen Ma, et al. Hybridnorm: Towards stable and efficient transformer training via hybrid normalization. *Advances in Neural Information Processing Systems*, 38:25106–25147, 2026.

APPENDIX

A MESSAGE FAILURE TEST

Communication in Dec-POMDPs has been studied under realistic constraints such as delays, failures, and costs Lauri et al. (2019). Here, we evaluate LC-MAPF under message transmission failures. For each agent and communication round, the updated message is replaced, with a given probability, by a random vector sampled from the standard normal distribution. We test LC-MAPF with 20% and 50% message failure on Random maps from the POGEMA benchmark and compare it with the original model. The results are presented in Table 6.

The success rate comparison reveals the negative impact of random noise messages compared to the LC-MAPF for 48 agents and above, highlighting the importance of the information conveyed in LC-MAPF messages for achieving performance improvement. However, despite the extreme setup with 50% message failure probability, agents can successfully solve simpler tasks with up to 32 agents, and obtain partial success in cases with larger agent populations.

	0% failure		20% failure		50% failure	
Agents	Success	Collisions	Success	Collisions	Success	Collisions
8	1.00 ±0.00	0.49 ±0.22	1.00 ±0.00	0.96 ±0.30	1.00 ±0.00	1.62 ±0.44
16	1.00 ±0.00	1.89 ±0.38	1.00 ±0.00	5.49 ±1.69	1.00 ±0.00	10.27 ±1.85
24	1.00 ±0.00	6.88 ±1.65	1.00 ±0.00	12.62 ±1.87	1.00 ±0.00	29.60 ±5.32
32	1.00 ±0.00	18.30 ±4.71	1.00 ±0.00	30.98 ±4.99	1.00 ±0.00	80.79 ±18.14
48	1.00 ±0.00	71.66 ±14.07	0.98 ±0.03	138.20 ±34.15	0.94 ±0.04	300.72 ±61.50
64	0.98 ±0.02	219.23 ±46.80	0.93 ±0.04	431.27 ±88.40	0.77 ±0.07	820.64 ±139.01

Table 6: Communication Failure Results on `Random` Maps. Zero message failure means the original LC-MAPF performance. The provided values are average ± 95% confidence interval. Best results are marked with background color per row.

B COMPARISON WITH REACTIVE SEARCH-BASED METHODS

We additionally compare LC-MAPF with two search-based reactive approaches: Priority Inheritance with Backtracking (PIBT) Okumura et al. (2022) and Real-Time LaCAM (RT-LaCAM) Liang et al. (2025). Unlike conventional full-horizon MAPF solvers, these methods do not need to construct complete paths for all agents or finish a search from the initial configuration to the goal configuration before execution begins. Instead, similarly to LC-MAPF, they select the agents' next actions online at each environment step. PIBT constructs a one-step joint move through priority inheritance and backtracking, whereas RT-LaCAM incrementally reuses its search state between planning and execution steps.

Fig. 6 shows that LC-MAPF consistently achieves substantially higher success rates as the number of agents increases, while also producing solutions with lower relative SoC. This experiment isolates the online, step-by-step decision-making paradigm from the conventional full-horizon planning setting. Under the evaluated benchmark and protocol, the existing reactive search-based approaches cannot match the performance of a learned policy such as LC-MAPF, demonstrating the benefit of learning coordinated local decision making directly from expert demonstrations.

C ADDITIONAL ROBOTICS DETAILS

C.1 PLATFORM AND SOFTWARE

The physical environment is designed as a modular arena composed of standardized floor tiles and wall segments, allowing for rapid reconfiguration of the maze layout. Each floor tile is a 30 cm × 30 cm square plywood module with pre-drilled holes that support the attachment of wall elements. Each agent is a modified Waveshare JetBot platform (Fig. 7, right) with dimensions 17cm × 17 cm × 22 cm. A Jetson Nano 4GB running Ubuntu 20.04 serves as the compute unit. The robot features DC motors with encoders and an RPLIDAR A1 scanner for mapping and obstacle detection. Communication uses the Zenoh plugin with CycloneDDS as the ROS2 rmw. SLAM

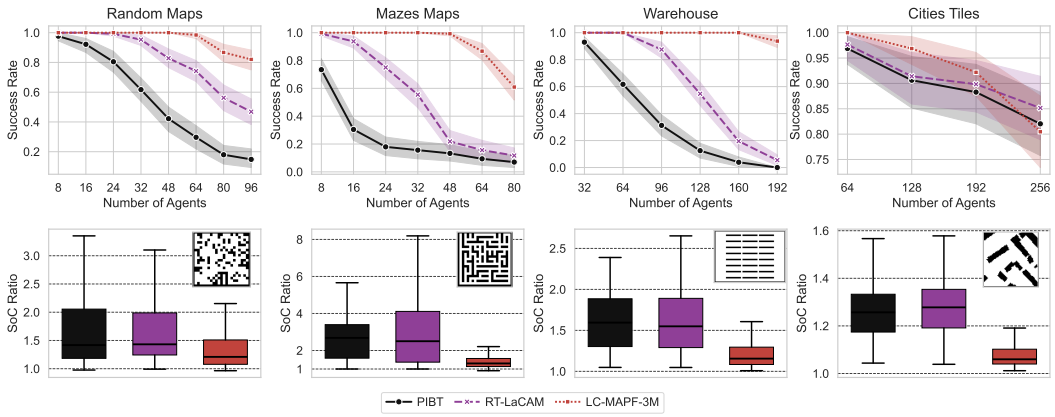


Figure 6: Additional comparison of LC-MAPF with the reactive search-based PIBT and RT-LaCAM approaches on the POGEMA benchmark. The top row shows success rates with 95% confidence intervals (higher is better); the bottom row shows the SoC ratio relative to LaCAM* solutions (lower is better).

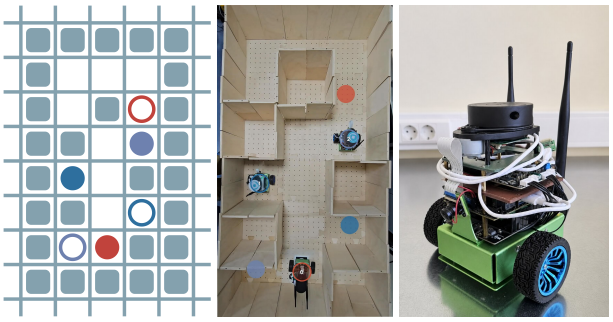


Figure 7: Real-world execution of maze scenario with 3 robots. Left: simulated POGEMA environment. Center: respective physical setup for real-world deployment. Right: deeply modified robotic agent based on Jetbot AI Kit.

Toolbox builds 2D occupancy grids, while Nav2 handles navigation using ThetaStar planner and Vector pursuit controller. Each robot runs its own copy of the policy and executes the four communication rounds as actual ROS 2 message exchanges transported through Zenoh. A host process is used only to synchronize robot positions by collecting each robot's current cell and redistributing a consistent team snapshot; it does not perform neural-network inference, message generation, or action selection.

C.2 Bridging Discrete Planning and Continuous Multi-Robot Execution

Several challenges arise when deploying LC-MAPF on real robots. First, we bridge the discrete grid and continuous space by overlaying a grid on the occupancy map and using contour detection to compute transformations between coordinate frames. Second, to handle differential-drive constraints versus holonomic assumptions, we decompose each step into synchronized rotation and translation phases. Third, we remove other robots from local costmaps to prevent conflicts with the global plan, delegating collision avoidance to the planner. Finally, under centralized control with a static environment, the map comes from simulation while robot poses are obtained from the TF tree.

C.3 Experimental Protocol

The physical maze replicated the simulation layout. Fig. 7 (left, center) shows a POGEMA scenario alongside its real-world setup, using identical initial positions and goals. The LC-MAPF policy controlled agents without modification.

Table 7: Values of key hyperparameters.

Parameter	**Value**
Learning rate schedule	Cosine decay
Maximum learning rate	6×10^{-4}
Minimum learning rate	6×10^{-5}
Warm-up iterations	8000
Gradient accumulation steps	16
Batch size	32
Block size	256
Encoder/Decoder layers	3
Attention heads	3
Embedding dimension (d_{embd})	192
Latent dimension (d_{latent})	96
Number of latent tokens (T_{latent})	32
Number of communication rounds	4

The recorded behaviors demonstrate effective real-world coordination and successful goal achievement, as shown in the accompanying video materials.

D Training Infrastructure and Hyperparameters

Training was performed for 800,000 iterations using a single NVIDIA H100 GPU with 80GB memory. On each iteration a local batch of 32 samples was used with 16 gradient accumulation steps, resulting in an effective batch size of 512 samples per optimization step. The total training time amounted to roughly 900 GPU-hours. Mixed-precision training (bfloat16) was used to improve throughput and reduce memory footprint, and all runs were executed using PyTorch 2.3.1 with CUDA 12.2.

Table 7 lists the key architectural and optimization hyperparameters.