# Learning to Communicate Locally for Large-Scale Multi-Agent Pathfinding

**Valeriy Vyaltsev, Alsu Sagirova, Anton Andreychuk, Yuri Kuratov, Konstantin Yakovlev, Aleksandr Panov, Alexey Skrynnik**

CogAI Lab
Moscow, Russia
andreychuk@cogailab.com

## 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 Dec-POMDP when 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 foundation 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 the scalability LC-MAPF, which is a common bottleneck for communication-based MAPF solvers.

## Introduction

Modern robotic systems often involve multiple mobile agents that must navigate and operate within shared environments, such as robots transporting goods in automated warehouses (Li et al. 2021a) or autonomous vehicles on public roads (Li et al. 2023). A key abstraction for modeling and solving the challenge of coordinating such agents safely is multi-agent pathfinding (MAPF) (Stern et al. 2019).

In MAPF, time is divided into discrete steps, and agents move on a graph structure (typically a 4-connected grid). Each agent acts synchronously, with each action, either moving to a neighboring vertex or waiting in place, taking exactly one time step. The goal is to compute a set of individual plans, one for each agent, that ensures no collisions occur as the agents execute them.



Many challenges of real-world robotics are not directly captured by the MAPF abstraction, including continuous space and time, asynchronous agent behavior, limited communication and observation, and various perception constraints. Despite these simplifications, MAPF successfully models the central difficulty in multi-robot navigation: coordinating agents to avoid collisions while aiming to optimize a specific cost function. As a result, MAPF has attracted substantial interest from both the robotics and AI research communities. Furthermore, a number of studies have demonstrated the successful application of MAPF-based methods to the continuous, noisy, and uncertain environments faced by real-world robotic systems (Hönig et al. 2016; Ma et al. 2019b).

MAPF is most commonly approached in a centralized setting, where a single planner with full knowledge of the environment is responsible for generating plans for all agents. A wide range of both optimal and suboptimal centralized solvers have been proposed (Standley 2010; Sharon et al. 2015; Wagner and Choset 2011; Surynek et al. 2016; Okumura et al. 2022; Okumura 2023; Li et al. 2022; Veerapaneni et al. 2024; Wang et al. 2025).

It is well established that optimal MAPF solvers scale poorly with the increasing numbers of agents, as the problem is NP-hard (Surynek 2010). Suboptimal solvers, on the other hand, can scale to thousands of agents, but their solution quality may degrade significantly. Consequently, a central focus of MAPF research is striking the balance between computational efficiency and solution quality.

One promising strategy for addressing this challenge is to adopt a decentralized approach. Here, MAPF is modeled as a decentralized sequential decision-making problem, where each agent independently selects and executes actions at every time step based on local observations. The resulting decision-making policy may be fully learned or designed as a hybrid, combining learnable and fixed components (Liu et al. 2020; Li et al. 2021b; Wang et al. 2023; Ma, Luo, and Ma 2021; Ma, Luo, and Pan 2021; Tang, Berto, and Park 2024; Skrynnik et al. 2024, 2023; Sagirova, Kuratov, and Burtsev 2025; Phan, Phan, and Koenig 2025). A recent survey provides a comprehensive overview of developments in this area (Alkazzi and Okumura 2024).

One of the recent advancements in decentralized, learnable MAPF is MAPF-GPT (Andreychuk et al. 2025b),

which relies entirely on supervised learning using a transformer-based neural network trained on an extensive dataset of approximately one billion observation-action pairs. Despite its simplicity, MAPF-GPT outperforms most of the state-of-the-art learning-based MAPF methods.

However, a major limitation of MAPF-GPT is its lack of agent-to-agent communication. While it learns collaborative behavior through the data, it does so without any communication between agents, as the training data is generated by a centralized solver that does not include communication signals. This means that MAPF-GPT cannot explicitly facilitate interaction or collaboration between agents during problem-solving, which could be a key factor in improving performance.

Several existing decentralized MAPF methods, such as SCRIMP, PICO, DCC, and DHC use communication mechanism. However, it is mostly limited to sharing local observations or internally known state information in one round of communication (Alkazzi and Okumura 2024). These mechanisms often fall short of enabling agents to engage in more meaningful coordination.

We argue that effective communication in decentralized MAPF should extend beyond single-message exchange and involve multiple rounds of agent interaction. Such iterative communication enables agents to negotiate, resolve conflicts, and build consistent joint plans that are crucial for robust multi-agent coordination in complex environments. Motivated by this, we explore how to equip a large transformer-based imitation learning model with the ability to perform effective local communication.

Our main contributions are the following:

- We introduce a novel communication learning framework called **LC-MAPF**, which enables agents to communicate using only the expert demonstrations of selected actions, without requiring explicit communication supervision.
- 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. We conduct extensive evaluations and compare it with existing learning-based approaches.
- Additionally, we extensively study how the number of communication rounds influences the performance of the agents, as shown in the ablation study. Moreover, we show that despite incorporating communication, the proposed mechanism maintains linear scalability as the number of agents grows.

## Related Work

The related work section covers three categories relevant to the proposed approach: foundation models for multi-agent systems, communication-based learning in MAPF, and multi-agent pathfinding.

### Foundation Models for Multi-Agent Systems

Foundation models are typically trained on large-scale datasets, enabling generalization through zero-shot or few-shot learning (Bommasani et al. 2021; Yang et al. 2023). 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; Ruoss et al. 2024), 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; Team et al. 2024; Kim et al. 2024), 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).

### 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; Sharon, Stern, and Goldenberg 2013; Wagner and Choset 2011), apply graph search techniques and often offer optimality or bounded-suboptimality guarantees. Simpler methods like prioritized planning (Ma et al. 2019a) trade off optimality for efficiency and scalability.

### 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 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, Luo, and Ma 2021) that demonstrate significant improvement over PRIMAL. Subsequent methods such as DCC (Ma, Luo, and Pan 2021) 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 example of a decentralized communication approach coming from the online MAPF is the SRMT (Sagirova, Kuratov, and Burtsev 2025). It allows agents to implicitly exchange information by generating and broadcasting agents' working memory representations learned by the memory-augmented transformer (Burtsev et al. 2020). The memory states used

for communication, are updated recurrently (Bulatov, Kuratov, and Burtsev 2022) to preserve the historical information and improve agents coordination.

# Background

## 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. Totally $n$ agents ($\mathcal{A} = \{u_1, \ldots, u_n\}$) are presented in the environment. The task is to find a set of plans $Pl = \{pl^i\}$ composed of the actions that can be either move to an adjacent vertex or stay at the current vertex. Additionally, 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 either 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 and remains there.

MAPF can also be formulated as a sequential decision-making problem, where the task is to construct a centralized policy $\pi_{\text{centralized}}$ that selects a joint, conflict-free action $\mathbf{a} = a^1 \times \cdots \times a^n$ at each timestep, with $a^i$ denoting agent $i$'s action. Such a policy can be hand-crafted or learned.

Finally, MAPF can also be treated as a decentralized decision-making problem where the goal is to learn a homogeneous individual policy $\pi$ shared by all agents, which selects an action for each agent based solely on local observations and, possibly, communication. The observations typically include information about nearby obstacles and agents, rather than the full global state.

## Imitation Learning for MAPF

Imitation learning seeks to approximate an expert policy $\hat{\pi}$ by training a parameterized policy $\pi_\theta$. A dataset $\mathcal{T}$ of expert trajectories is first collected: $\hat{\mathcal{T}} = \{\hat{\tau}_i\}_{i=1}^{K}$, where each $\hat{\tau}_i = \{(s^1, \mathbf{a}^1), \ldots, (s^L, \mathbf{a}^L)\}$ 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 $\tau_u^{\hat{\pi}} = \{(o_u^1, a_u^1), \ldots, (o_u^L, a_u^L)\}$ 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 (e.g., in transformer-based models (Ruoss et al. 2025)). 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:

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

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

# Method

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

$$o_u^t = [\text{cost-to-go}_u^t,\ i_u^t,\ n_{u,1}^t, \ldots, n_{u,k}^t], \quad (2)$$

where $\text{cost-to-go}_u^t$ 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.

The observation is converted into a sequence of embeddings:

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

where $E_{\text{tok}}$ encodes the token identity, $E_{\text{pos}}$ provides a positional index inside the sequence, and $E_{\text{nbr}}$ 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:

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

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 the encoded tokens and produces a compact latent state:

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

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. The agents then perform $R_{\text{comm}}$ 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 messages from neighbors:

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

where $E_{\text{nbr}}^{dec}$ indicates which agent produced each message. Messages are inserted into a decoding module together with the agent's latent state:

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

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

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

After the final round, a prediction head produces the action logits:

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

Our Transformer blocks integrate several recent advancements aimed at improving stability and performance, including RMSNorm (Zhang and Sennrich 2019) for normalization, SwiGLU (Shazeer 2020) feed-forward layers, a combined pre- and post-normalization and QK-normalization

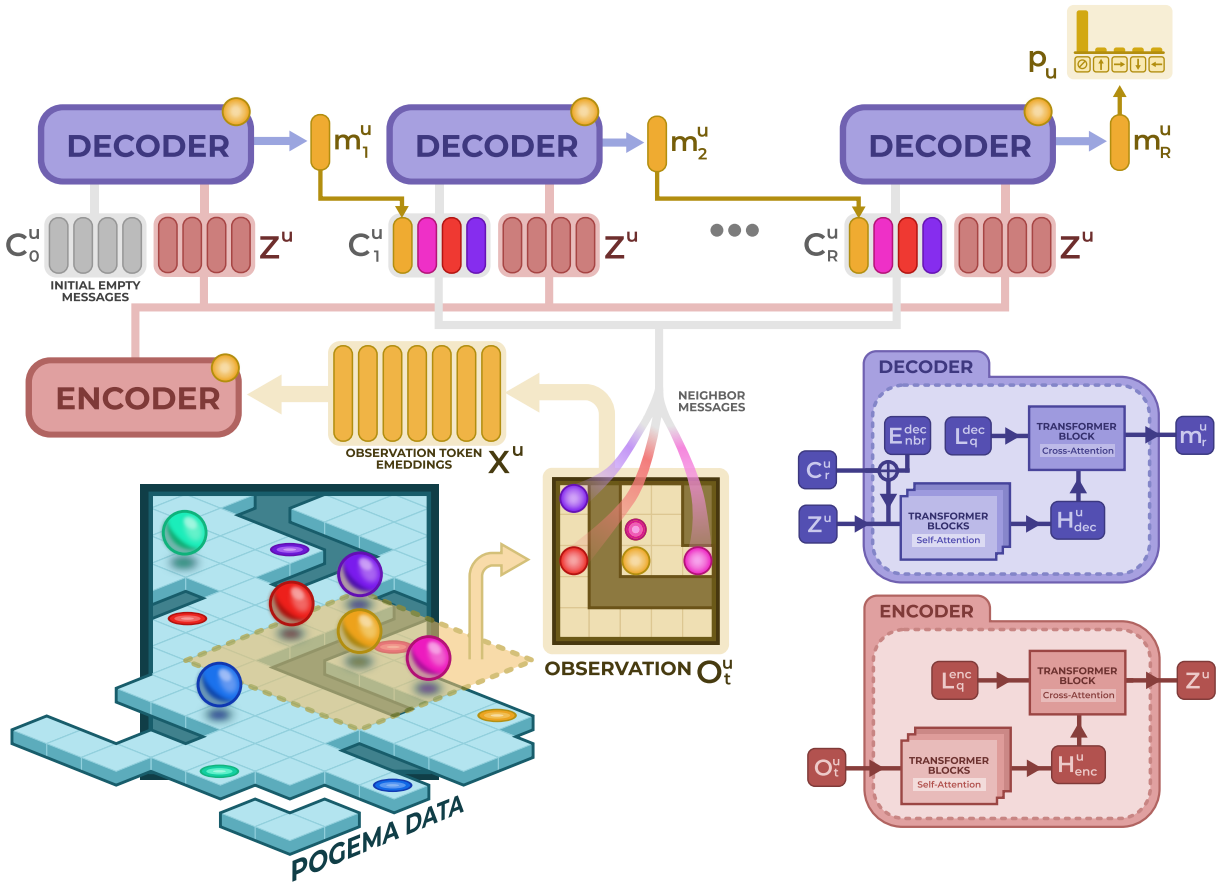


Figure 1: Overview of the proposed LC-MAPF architecture. Each agent $u \in \mathcal{U}$ encodes its local observation $o_u^t$ into a latent representation $z_u$ using a Transformer-based encoder.The latent state participates in an iterative message-passing procedure over $R$ communication rounds. At each round $r$, the agent receives the set of neighbor messages $C_u^r$ and fuses them with its latent state $z_u$ through a Transformer-based decoder, producing an updated message $m_u^r$ that will be sent to its neighbors in the next round. After $R$ communication rounds, the decoder outputs action logits $a_u$, which are converted to an action probability distribution $p_u = \mathrm{softmax}(a_u)$. Both encoder and decoder consist of stacked Transformer blocks with self-attention and cross-attention: the encoder integrates the tokenized observation $o_u^t$, while the decoder integrates the agent's latent representation $z_u$ with contextualized neighbor messages $C_u^r$, enabling decentralized coordination through end-to-end learned communication.

scheme (Zhuo et al. 2025), and a differential attention mechanism (Ye et al. 2025).

LC-MAPF is trained from expert demonstrations in a fully end-to-end manner. The training objective is the cross-entropy loss:

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

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 network learns what information should be communicated and communication can emerge naturally from optimization of the shared objective. In all our experiments we use $R_{\text{comm}} = 4$ communication rounds.

## Experimental Setup

The experiments were conducted on the **POGEMA** benchmark (Skrynnik et al. 2025a), which provides a diverse set of partially observable multi-agent pathfinding (MAPF) environments, including *Random*, *Mazes*, *Warehouse*, *Cities*, and *Puzzles* map types. Each agent receives a tokenized observation of up to **256 tokens**, corresponding to its $11 \times 11$ local field of view, agent-specific attributes, and spatial context. Agents can communicate with up to 13 nearby agents within a 5-cell radius, and neighboring agents are ordered by their Manhattan distance to the ego-agent, ensuring consistent positional ordering in the communication graph.

The training dataset consisted of aggregated samples from three subsets of the POGEMA benchmark: *mazes*, *random*, and *house* maps. The combined dataset contained approxi-

mately **23.5 million samples** with a 0.6:0.2:0.2 distribution across the three subsets. In contrast to the dataset used for training the original MAPF-GPT, in this dataset each sample contains observations and ground-truth actions for all agents, instead of a particular one. Thus, the total number of observation-action pairs is roughly **750 million**. Additionally to the observations and ground-truth actions, the dataset contains adjacency information describing the local communication structure.

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 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.

The model contained approximately **3 million trainable parameters** and was trained from scratch using the AdamW optimizer with cosine learning rate decay. Key architectural and optimization hyperparameters are summarized in Table 1.

Table 1: Values of key hyperparameters.

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

## 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, Luo, and Pan 2021), and EPH (Tang, Berto, and Park 2024). The original MAPF-GPT comes in three different sizes: 2M, 6M, and 85M. In our experiments, we used only the largest and best performing model with 85M parameters. The experiments were conducted on the POGEMA benchmark (Skrynnik et al. 2025b) with the same evaluation protocol as in the original MAPF-GPT paper. Specifically, we used 4 map types: Random, Mazes, Warehouse, and Cities Tiles. The first two are the same type of maps that were used to train LC-MAPF, the latter two differ significantly in topology and are used to evaluate the ability to generalize to out-of-distribution map types. Mazes and Random maps range in size from $17 \times 17$ to $21 \times 21$, and contain up to 64 agents. The Warehouse type features a single map of size $33 \times 46$ with restrictions on where start and goal locations can be placed (to model real-world fulfillment scenarios). The maximum number of agents on this map is 192. The Cities Tiles maps are of $64 \times 64$ 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. 2025b)

We also performed two additional series of experiments. One studied the influence of number of communication rounds on the overall performance, while the other one is conducted to demonstrate that addition of communication does not violate one of the core features of MAPF-GPT - linear scalability to the number of agents.

### Comparison with the Baselines

The first series of experimental results is shown in Fig. 2 and Fig. 3. In the former, the average success rate for each map type is presented based on the number of agents in the instances. As can be seen, in all the cases LC-MAPF is either on par or better than any baseline, including original MAPF-GPT with a huge 85M model and fine-tuned MAPF-GPT-DDG. Comparing with other competitors, SCRIMP demonstrates the best performance among baselines. However, LC-MAPF is able to outperform it on this map as well.

Fig. 3 presents the ratio of SoC (solution cost) relative to the solution found by the centralized planner, LaCAM*, in the form of box-and-whiskers plots. These results align with those presented in Fig. 2 and show that LC-MAPF achieves the best results in most of the cases. Surprisingly, while MAPF-GPT-DDG has a worse success rate than LC-MAPF on Warehouse map, its average solution cost is slightly better.

### Ablation study

During the ablation study of LC-MAPF we wanted to investigate the influence of the communication mechanism on the performance of the approach. To this end, we varied the number of communication rounds employed by LC-MAPF from 1 to 8. The experiments were conducted on `Warehouse` map with number of agents varying from 32 to 192. For each number of agents all 128 testing instances provided by the POGEMA Behcnmark (Skrynnik et al. 2025b) were used. The length of the episode was set to 128. Two performance indicators were tracked: success rate (the ratio of the successfully solved instances) and number of collisions. The results are shown in Table 2.

The obtained results clearly demonstrates that a) LC-MAPF needs at least 2 rounds of communication to solve at least some of the instances; b) the best performance LC-MAPF demonstrates with the number of rounds that was

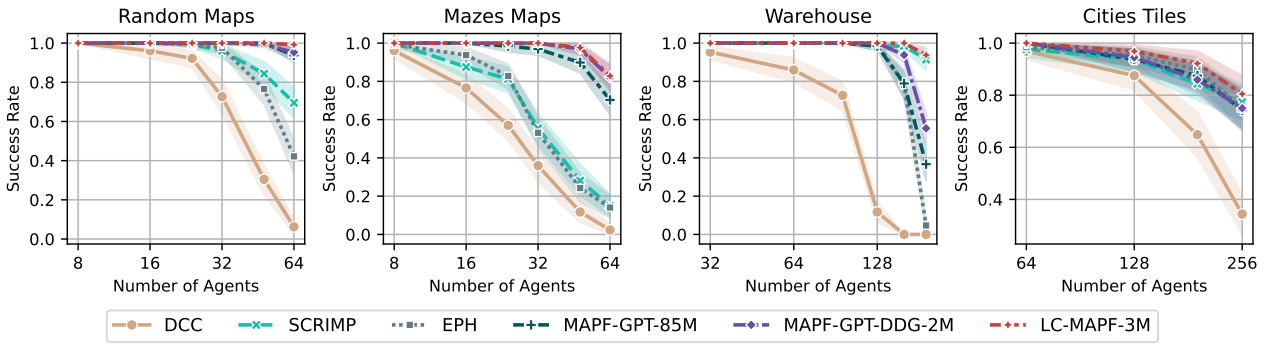


Figure 2: Success rates of the approaches on different map types depending on the number of agents in the instances (higher is better). The shaded area indicates the 95% confidence interval.

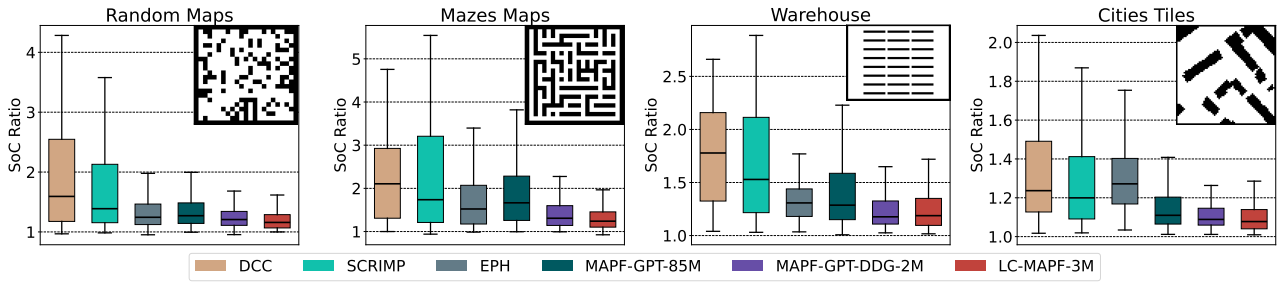


Figure 3: SoC ratio relative to solutions found by the LaCAM* approach (lower is better).

| | Success rate across different LC-MAPF communication rounds | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Agents | Rounds=1 | Rounds=2 | Rounds=3 | Rounds=4 | Rounds=5 | Rounds=6 | Rounds=7 | Rounds=8 |
| 32 | 0.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 |
| 64 | 0.000 ± 0.000 | 0.766 ± 0.070 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 |
| 96 | 0.000 ± 0.000 | 0.094 ± 0.051 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 |
| 128 | 0.000 ± 0.000 | 0.008 ± 0.012 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 |
| 160 | 0.000 ± 0.000 | 0.016 ± 0.020 | 0.984 ± 0.020 | 1.000 ± 0.000 | 1.000 ± 0.000 | 1.000 ± 0.000 | 0.992 ± 0.012 | 1.000 ± 0.000 |
| 192 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.742 ± 0.074 | 0.938 ± 0.043 | 0.914 ± 0.051 | 0.883 ± 0.055 | 0.938 ± 0.043 | 0.938 ± 0.043 |
| | Collision counts across different LC-MAPF communication rounds | | | | | | | |
| Agents | Rounds=1 | Rounds=2 | Rounds=3 | Rounds=4 | Rounds=5 | Rounds=6 | Rounds=7 | Rounds=8 |
| 32 | 299.047 ± 6.149 | 61.961 ± 2.794 | 31.977 ± 1.539 | 5.438 ± 0.743 | 8.289 ± 1.012 | 8.055 ± 0.727 | 6.664 ± 0.754 | 7.266 ± 0.782 |
| 64 | 787.773 ± 7.359 | 258.078 ± 6.611 | 127.977 ± 3.669 | 24.914 ± 1.633 | 36.164 ± 2.379 | 42.250 ± 2.012 | 32.750 ± 2.004 | 30.188 ± 1.965 |
| 96 | 1361.750 ± 14.048 | 618.430 ± 16.111 | 289.789 ± 9.089 | 79.484 ± 4.542 | 112.781 ± 6.368 | 119.922 ± 5.572 | 105.938 ± 5.770 | 92.641 ± 5.208 |
| 128 | 2077.656 ± 26.500 | 1144.828 ± 46.349 | 564.547 ± 17.383 | 226.641 ± 13.798 | 294.492 ± 14.356 | 291.977 ± 13.338 | 264.773 ± 14.460 | 252.688 ± 12.756 |
| 160 | 3317.375 ± 90.149 | 2107.891 ± 81.585 | 1057.641 ± 44.313 | 486.711 ± 23.802 | 641.992 ± 38.691 | 639.562 ± 31.303 | 585.219 ± 28.836 | 570.234 ± 31.782 |
| 192 | 5330.273 ± 142.768 | 3556.984 ± 136.774 | 2038.875 ± 107.579 | 1175.117 ± 77.500 | 1422.742 ± 74.590 | 1408.836 ± 79.931 | 1253.203 ± 72.768 | 1224.867 ± 67.736 |

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. Tan boxes highlight the best mean values for visibility purposes.

used during training, i.e. 4; c) further increment of communication rounds does not lead to improvement of metrics, however, comparing them between each other demonstrates that higher number of communications round reduces the number of collisions that occurs between agents.

## Scalability Analysis

To better demonstrate the superior scalability of LC-MAPF, we present the actual decision times of all evaluated learning-based approaches with communication capabilities: DCC, SCRIMP, and LC-MAPF. Table 3 shows the average time required for all agents to determine their next action across varying numbers of agents in the `Warehouse` map scenarios.

| Algorithm | 32 agents | 64 agents | 128 agents | 192 agents |
|---|---|---|---|---|
| DCC | 48.0 ± 1.0 | 164.0 ± 2.0 (×3.4) | 619.0 ± 2.0 (×12.9) | 1314.0 ± 3.0 (×27.4) |
| SCRIMP | 47.0 ± 1.0 | 106.0 ± 1.0 (×2.3) | 388.0 ± 7.0 (×8.3) | 1190.0 ± 25.0 (×25.3) |
| LC-MAPF | 19.8 ± 0.5 | 29.4 ± 0.2 **(×1.5)** | 54.1 ± 0.4 **(×2.7)** | 78.4 ± 0.5 **(×4.0)** |

Table 3: Decision time (in milliseconds) in the instances with different numbers of agents on `Warehouse` map.

LC-MAPF demonstrates superior efficiency, scaling even better than linear. Such scaling is explained by the ability of GPU to process agents in parallel with a larger batch

size. While LC-MAPF performs multiple rounds of communication, it requires less computation time than DCC or SCRIMP. In contrast, when handling 192 agents, SCRIMP and DCC require 25.3 and 27.4 times more computation time, respectively, compared to their performance with 32 agents.

## Conclusion

We introduced LC-MAPF, a novel communication learning framework for decentralized multi-agent pathfinding that leverages expert demonstrations without explicit communication supervision. The communication is organized in rounds to increase the level of cooperation between the agents. Our transformer-based model outperforms state-of-the-art learning-based MAPF solvers on the POGEMA benchmark, improving coordination and cooperation 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 foundation model that offers an effective and scalable solution for decentralized multi-agent pathfinding through multi-round local communication.

## References


Alkazzi, J.-M.; and Okumura, K. 2024. A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. *IEEE Access*.

Andreychuk, A.; Yakovlev, K.; Panov, A.; and Skrynnik, A. 2025a. Advancing Learnable Multi-Agent Pathfinding Solvers with Active Fine-Tuning. *arXiv preprint arXiv:2506.23793*.

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

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dkebiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Bulatov, A.; Kuratov, Y.; and Burtsev, M. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35: 11079–11091.

Burtsev, M. S.; Kuratov, Y.; Peganov, A.; and Sapunov, G. V. 2020. Memory transformer. *arXiv preprint arXiv:2006.11527*.

Firoozi, R.; Tucker, J.; Tian, S.; Majumdar, A.; Sun, J.; Liu, W.; Zhu, Y.; Song, S.; Kapoor, A.; Hausman, K.; et al. 2023. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*.

Fourney, A.; Bansal, G.; Mozannar, H.; Tan, C.; Salinas, E.; Niedtner, F.; Proebsting, G.; Bassman, G.; Gerrits, J.; Alber, J.; et al. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

Hönig, W.; Kumar, T. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-Agent Path Finding with Kinematic Constraints. In *Proceedings of The 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 477–485.

Jaegle, A.; Borgeaud, S.; Alayrac, J.-B.; Doersch, C.; Ionescu, C.; Ding, D.; Koppula, S.; Zoran, D.; Brock, A.; Shelhamer, E.; Hénaff, O.; Botvinick, M. M.; Zisserman, A.; Vinyals, O.; and Carreira, J. 2022. Perceiver IO: A General Architecture for Structured Inputs & Outputs. arXiv:2107.14795.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E. P.; Sanketi, P. R.; Vuong, Q.; Kollar, T.; Burchfiel, B.; Tedrake, R.; Sadigh, D.; Levine, S.; Liang, P.; and Finn, C. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. In *8th Annual Conference on Robot Learning*.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10256–10265.

Li, J.; Lin, E.; Vu, H. L.; Koenig, S.; et al. 2023. Intersection coordination with priority-based search for autonomous vehicles. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, 11578–11585.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. S.; and Koenig, S. 2021a. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, 11272–11281.

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

Liu, Z.; Chen, B.; Zhou, H.; Koushik, G.; Hebert, M.; and Zhao, D. 2020. Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020)*, 11748–11754. IEEE.

Ma, H.; Harabor, D.; Stuckey, P. J.; Li, J.; and Koenig, S. 2019a. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 7643–7650.

Ma, H.; Hönig, W.; Kumar, T. K. S.; Ayanian, N.; and Koenig, S. 2019b. Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, 7651–7658.

Ma, Z.; Luo, Y.; and Ma, H. 2021. Distributed heuristic multi-agent path finding with communication. In *2021*

*IEEE International Conference on Robotics and Automation (ICRA 2021)*, 8699–8705. IEEE.

Ma, Z.; Luo, Y.; and Pan, J. 2021. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters*, 7(2): 1455–1462.

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

Okumura, K. 2024. 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*, 1501–1509.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310: 103752.

Phan, T.; Phan, T.; and Koenig, S. 2025. Generative Curricula for Multi-Agent Path Finding via Unsupervised and Reinforcement Learning. *Journal of Artificial Intelligence Research*, 82: 2471–2534.

Ruoss, A.; Delétang, G.; Medapati, S.; Grau-Moya, J.; Li, K.; Catt, E.; Reid, J.; Lewis, C.; Veness, J.; and Genewein, T. 2025. Amortized planning with large-scale transformers: A case study on chess. *Advances in Neural Information Processing Systems*, 37: 65765–65790.

Ruoss, A.; Deletang, G.; Medapati, S.; Grau-Moya, J.; Wenliang, L. K.; Catt, E.; Reid, J.; Lewis, C. A.; Veness, J.; and Genewein, T. 2024. Amortized Planning with Large-Scale Transformers: A Case Study on Chess. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sagirova, A.; Kuratov, Y.; and Burtsev, M. 2025. SRMT: Shared Memory for Multi-agent Lifelong Pathfinding. arXiv:2501.13200.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. S.; Koenig, S.; and Choset, H. 2019. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3): 2378–2385.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219: 40–66.

Sharon, G.; Stern, R.; and Goldenberg, A., Meir aand Felner. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence*, 195: 470–495.

Shazeer, N. 2020. GLU Variants Improve Transformer. arXiv:2002.05202.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Skrynnik, A.; Andreychuk, A.; Borzilov, A.; Chernyavskiy, A.; Yakovlev, K.; and Panov, A. 2025a. POGEMA: A Benchmark Platform for Cooperative Multi-Agent Pathfinding. arXiv:2407.14931.

Skrynnik, A.; Andreychuk, A.; Borzilov, A.; Chernyavskiy, A.; Yakovlev, K.; and Panov, A. 2025b. POGEMA: A Benchmark Platform for Cooperative Multi-Agent Pathfinding. In *The Thirteenth International Conference on Learning Representations*.

Skrynnik, A.; Andreychuk, A.; Nesterova, M.; Yakovlev, K.; and Panov, A. 2024. Learn to Follow: Decentralized Lifelong Multi-agent Pathfinding via Planning and Learning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*.

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

Standley, T. S. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of The 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 173–178.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. S.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS 2019)*, 151–158.

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

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. 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)*, 810–818. IOS Press.

Tang, H.; Berto, F.; and Park, J. 2024. Ensembling prioritized hybrid policies for multi-agent pathfinding. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8047–8054. IEEE.

Team, O. M.; Ghosh, D.; Walke, H.; Pertsch, K.; Black, K.; Mees, O.; Dasari, S.; Hejna, J.; Kreiman, T.; Xu, C.; et al. 2024. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*.

Veerapaneni, R.; Wang, Q.; Ren, K.; Jakobsson, A.; Li, J.; and Likhachev, M. 2024. Improving learnt local MAPF policies with heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 597–606.

Wagner, G.; and Choset, H. 2011. M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of The 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, 3260–3267.

Wang, Y.; Duhan, T.; Li, J.; and Sartoretti, G. 2025. 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, 23343–23350.

Wang, Y.; Xiang, B.; Huang, S.; and Sartoretti, G. 2023. SCRIMP: Scalable communication for reinforcement-and

imitation-learning-based multi-agent pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9301–9308. IEEE.

Yang, S.; Nachum, O.; Du, Y.; Wei, J.; Abbeel, P.; and Schuurmans, D. 2023. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Ye, T.; Dong, L.; Xia, Y.; Sun, Y.; Zhu, Y.; Huang, G.; and Wei, F. 2025. Differential Transformer. arXiv:2410.05258.

Zhang, B.; and Sennrich, R. 2019. Root Mean Square Layer Normalization. arXiv:1910.07467.

Zhuo, Z.; Zeng, Y.; Wang, Y.; Zhang, S.; Yang, J.; Li, X.; Zhou, X.; and Ma, J. 2025. HybridNorm: Towards Stable and Efficient Transformer Training via Hybrid Normalization. arXiv:2503.04598.