PAIRWISE IS NOT ENOUGH: HYPERGRAPH NEURAL NETWORKS FOR MULTI-AGENT PATHFINDING

Rishabh Jain[1], **Keisuke Okumura**[1,2], **Michael Amir**[1], **Pietro Liò**[1], **Amanda Prorok**[1]
[1]University of Cambridge, UK, [2]AIST, Japan
{rj412,ko393,ma2151,pl219,asp45}@cst.cam.ac.uk

ABSTRACT

Multi-Agent Path Finding (MAPF) is a representative multi-agent coordination problem, where multiple agents are required to navigate to their respective goals without collisions. Solving MAPF optimally is known to be NP-hard, leading to the adoption of learning-based approaches to alleviate the online computational burden. Prevailing approaches, such as Graph Neural Networks (GNNs), are typically constrained to *pairwise* message passing between agents. However, this limitation leads to suboptimal behaviours and critical issues, such as attention dilution, particularly in dense environments where group (i.e. beyond just two agents) coordination is most critical. Despite the importance of such higher-order interactions, existing approaches have not been able to fully explore them. To address this representational bottleneck, we introduce HMAGAT (Hypergraph Multi-Agent Attention Network), a novel architecture that leverages attentional mechanisms over directed hypergraphs to explicitly capture group dynamics. Empirically, HMAGAT establishes a new state-of-the-art among learning-based MAPF solvers: e.g., despite having just 1M parameters and being trained on 100× less data, it outperforms the current SoTA 85M parameter model. Through detailed analysis of HMAGAT's attention values, we demonstrate how hypergraph representations mitigate the attention dilution inherent in GNNs and capture complex interactions where pairwise methods fail. Our results illustrate that appropriate inductive biases are often more critical than the training data size or sheer parameter count for multi-agent problems.

1 INTRODUCTION

Multi-agent coordination is a fundamental topic in multi-agent systems, where multiple agents work together to achieve a common goal. Applications include warehouse automation (Wurman et al., 2008), autonomous driving (Shalev-Shwartz et al., 2016), traffic management (Adler & Blue, 2002) and manufacturing process control (Leng et al., 2023). These settings involve complex interactions and dependencies among agents. In the literature, such interactions are typically modelled in a pairwise manner, using graph neural networks (GNNs) or transformers (Bernárdez et al., 2023; Liu et al., 2024; Li et al., 2024b). However, pairwise interactions are insufficient to fully capture highly-coupled multi-agent dynamics, where interactions may involve multiple agents simultaneously.

This insufficiency has led to the exploration of higher-order representational structures, such as *hypergraphs*, which can naturally model group interactions. Recent works have shown that hypergraphs can effectively model small-scale multi-agent problems (Zhu et al., 2024; Zhao et al., 2025) and simple group interactions, like social groups in trajectory prediction tasks (Lin et al., 2024; Xu et al., 2022). However, the question remains, "Can hypergraphs scale beyond simple group settings to capture the dynamics of complex, highly-coupled multi-agent tasks?"

In this work, we provide a positive answer to this question by focusing on one such problem of *multi-agent pathfinding (MAPF)*, where the goal is to compute collision-free paths that efficiently guide a team of agents to their respective destinations. Optimally solving MAPF is known to be NP-hard (Surynek, 2010), even in sparse grids (Geft & Halperin, 2022), which highlights the strong entanglement among agents. This makes MAPF a suitable testbed for studying highly-coupled multi-agent problems, especially in obstacle-dense maps with large numbers of agents.

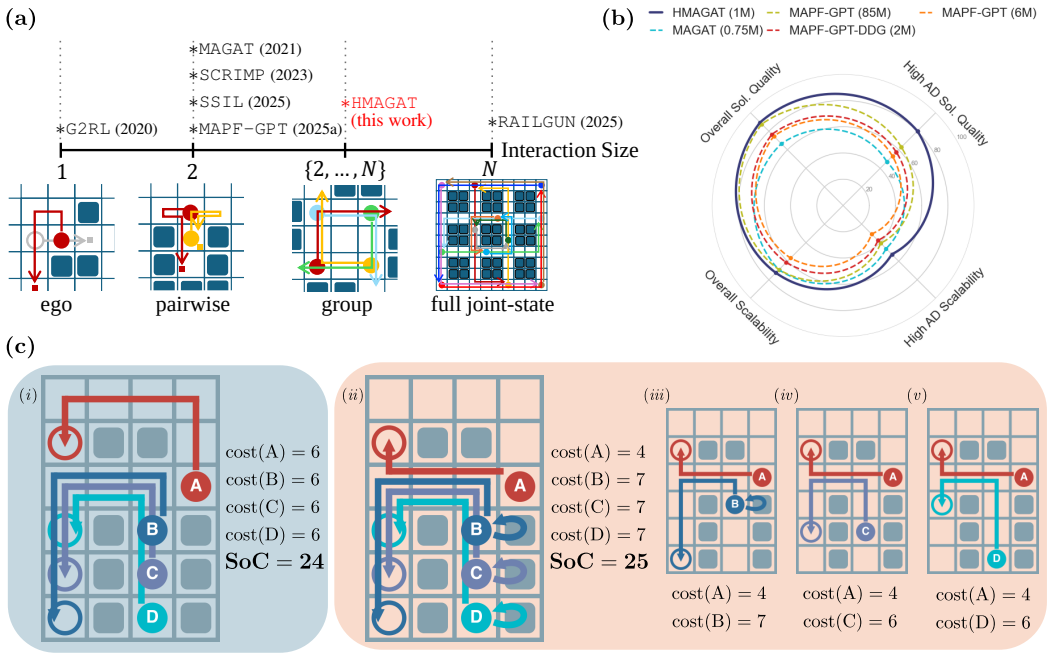

Figure 1: **(a)** Positioning of our work in relation to existing MAPF works with respect to the interaction modelling. **(b)** Radar plot comparing `HMAGAT` with other learning-based MAPF solvers, showing average solution quality and scalability (both the higher the better) across different small maps[1]and agent densities, and a focused view on the highest agent density (High AD) scenarios. Details of the metrics are in Appendix B. **(c)** MAPF requires group interactions. (*i*) The SoC-optimal (sum-of-costs) group interaction-based solution. (*iii-v*) Pairwise interactions with agent A, showing the optimal paths when agent pairs are isolated. (*ii*) Combination of the pairwise solutions. This solution is sub-optimal compared to the group interaction-based solution.

As with the other multi-agent problems, previous works on MAPF have focused on pairwise interaction-modelling, popularly using GNNs (Li et al., 2020; 2021; Veerapaneni et al., 2025) or transformers (Wang et al., 2023; Andreychuk et al., 2025a). However, MAPF is inherently a group problem, i.e., optimality and completeness can, generally, only be achieved when modelling the full joint state space of all agents (not just two), as shown in Figure 1b. We argue that the use of hypergraphs provides strong inductive biases for capturing group interactions, thereby enabling the development of more capable MAPF approaches.

To address this gap, *(i)* we propose `HMAGAT`, a novel imitation learning framework for MAPF, leveraging a hypergraph attention network to better model the higher-order group interactions. *(ii)* We propose hypergraph generation strategies that dynamically construct directed hypergraphs and *(iii)* empirically demonstrate the prowess of our approach over existing solvers, as shown in Fig. 1c, beating the previous state-of-the-art learning model while using only ~1.2% parameters and ~1% of the training data. *(iv)* Through experiments, we perform a deeper analysis on why pairwise attention fails while modelling group interactions, and how hypergraphs overcome these issues.

2 Preliminaries

2.1 Hypergraphs and Multi-Agent Learning

Hypergraphs are a higher-order generalisation of graphs, where edges, called *hyperedges*, can connect any number of nodes. Formally, a *directed hypergraph* is defined as $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where

[1]Large maps not included in the radar plot, as `MAPF-GPT` fails to scale to them, resulting in skewed results.

$\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed hyperedges. Each hyperedge $e \in \mathcal{E}$ is an ordered pair $(T(e), H(e))$, where $T(e) \subseteq \mathcal{V}$ is the tail and $H(e) \subseteq \mathcal{V}$ is the head of the hyperedge. A set of hyperedges incident to node $i \in \mathcal{V}$ is denoted as $\Gamma(i) = \{e \in \mathcal{E} \mid i \in T(e) \cup H(e)\}$.

The hypergraph counterpart of graph neural networks (GNNs) is referred to as *hypergraph neural networks (HGNNs)*. HGNNs have been increasingly adopted in multi-agent learning tasks due to their ability to capture higher-order interactions. E.g., HGNNs for formation control and robotic warehouses (Zhao et al., 2025), multi-agent continuous trajectory prediction (Lin et al., 2024; Xu et al., 2022) and social robot navigation (Li et al., 2024a; Wang et al., 2024). While these studies focus on problems with often modest team sizes ($\approx 10^1$ agents), where tightly coupled coordination among agents is less critical, our work explores the potential of HGNNs in a more challenging coupled setting containing large number of agents, namely, MAPF.

2.2 Multi-Agent Pathfinding (MAPF)

We consider a widely used MAPF formulation (Stern et al., 2019), defined by a team of agents $A = \{1, 2, \ldots, n\}$, a four-connected grid graph $G = (V, E)$, and a distinct start $s_i \in V$ and goal $g_i \in V$ for each agent $i \in A$. At each timestep, each agent either stays in place or moves to an adjacent vertex. Vertex and edge collisions are prohibited: agents cannot occupy the same vertex simultaneously or swap their occupied vertices. Then, for each agent $i \in A$, we aim to assign a collision-free path $\pi_i = (v^0 = s_i, v^1, \ldots, v^T = g_i)$. The solution quality is assessed by *sum-of-costs* (SoC), which sums the travel time of each agent until it stops at the target location. Herein, a *configuration* $\mathcal{Q} \in V^n$ refers to the locations for all agents. The dist function returns the shortest path length between two vertices on G, while $\mathsf{neigh}(v)$ represents adjacent vertices for $v \in V$.

MAPF is an appealing benchmark to assess the capabilities of multi-agent learning techniques, tailored to industrial setups such as warehouse automation (Agaskar et al., 2025). Several directions exist on how to leverage ML for MAPF (Alkazzi & Okumura, 2024), with a major thrust being the governance of agent-wise neural policies that interpret their surrounding information and output actions, ideally leading to coordination beyond mere greedy behaviour. Such policies are typically constructed either via reinforcement learning (RL) (Sartoretti et al., 2019) or imitation learning (IL) (Li et al., 2021). This work focuses on IL setups, motivated by recent findings that *(i)* collecting demonstration data is inexpensive with modern MAPF solvers (Okumura, 2024), and *(ii)* IL consistently outperforms RL (Andreychuk et al., 2025a; Veerapaneni et al., 2025).

2.3 `MAGAT`: GNN-based policy for MAPF

Among existing IL policies for MAPF to date, `MAGAT` (Li et al., 2021) is a representative approach that employs GNNs to capture pairwise agent interactions. Formally, it governs a policy $\pi_i(v \mid o_i, \mathcal{G})$ that outputs an action distribution over candidate target vertices v for agent i, given its observation o_i and the communication graph $\mathcal{G}$, which is constructed based on spatial proximity within a radius $R^{\text{comm}} \in \mathbb{N}_{>0}$. The model consists of *(i)* a CNN encoder that converts o_i into the node feature $\mathbf{x}_i$, *(ii)* a GNN layer that aggregates messages from other agents following $\mathcal{G}$, and *(iii)* an MLP decoder.

Our proposed `HMAGAT` builds upon `MAGAT`. In addition to replacing the GNN with HGNN layers, `HMAGAT` also incorporates techniques introduced in follow-up works on `MAGAT`. These include stacking GNN layers (three) to enhance representational power (Veerapaneni et al., 2025) and incorporating position-based edge features embedded into $\mathcal{G}$. Our implementation of `MAGAT` also includes these improvements.

Local Observation. Throughout the rest of the paper, we fix the observation tensor o_i for agent i at a configuration $\mathcal{Q}$ to be centred around its current position $\mathcal{Q}[i]$, with shape $4 \times (2R^{\text{obs}} + 1) \times (2R^{\text{obs}} + 1)$, where $R^{\text{obs}} \in \mathbb{N}_{>0}$ determines the FOV size. The four channels consist of *(i)* an obstacle map, *(ii)* an agent map, *(iii)* a projection of the goal direction, and *(iv)* a normalised cost-to-go map. For each location $v \in V$ within FOV, the normalised cost-to-go value is computed as $(\mathsf{dist}(v, g_i) - \mathsf{dist}(\mathcal{Q}[i], g_i))/(2R^{\text{obs}})$, while setting it to 1 for obstacles. This construction is commonly used in learning-based MAPF studies (Alkazzi & Okumura, 2024).

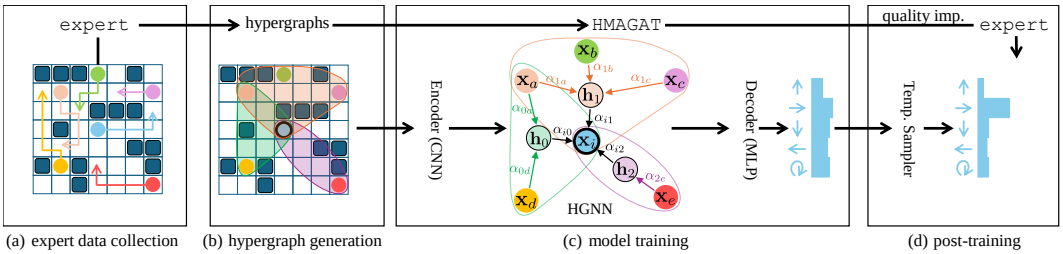

Figure 2: Overview of HMAGAT. (a) We collect demonstrations using an expert solver over 21K instances. (b) For each timestep, we extract a directed hypergraph representation (head agent shown with bold outline here). (c) These are used to train our HGNN-based model. (d) Post-training, we train the model using expert trajectories over intermediate instances, to improve solution quality. A temperature sampler is also trained to make the model more confident.

3 HMAGAT

While GNNs have shown promise in modelling multi-agent systems, they inherently focus on pairwise interactions between agents. This pairwise focus can limit their ability to capture the complex coupled-dynamics in multi-agent systems with interactions involving multiple agents simultaneously. We identify two key limitations of GNNs, specifically MAGAT, in this context.

Attention in dense scenarios. In multi-agent systems, agent interactions tend to be uneven, where for a given agent, some interactions might be more important than others. Thus, methods like MAGAT use an attentional mechanism to better model the interactions between agents. The attention scores are computed in a pairwise manner and normalised over the neighbourhood (via softmax). This, however, can be problematic in out-of-distribution dense scenarios. In such scenarios, each agent will have a dense neighbourhood, with only a few agents being truly relevant. However, since the attention scores are first computed pairwise and then normalised over the neighbourhood, the presence of many irrelevant agents will dilute the attention scores of the relevant agents.

Group Interactions. The interactions in GNNs are inherently pairwise, while multi-agent problems, like MAPF, tend to be fundamentally a group planning problem. Models like MAGAT can assuage this to some extent by using multiple GNN layers, where deeper layers can theoretically learn to capture group interactions. However, without an explicit bias for these group interactions, the model would likely struggle to learn such interactions.

Motivated by these observations, we propose HMAGAT, an attentional hypergraph neural network (HGNN)-based imitation learning model for MAPF. We first describe the architecture of HMAGAT, and then explore different hypergraph generation strategies.

3.1 Architecture

To model the group interaction of multiple agents influencing a single agent's decision, we design directed hypergraphs, with a singleton head and a multi-node tail. We replace the GNN layers in MAGAT with our HGNN layers, resulting in a pipeline consisting of *(i)* a CNN encoder, followed by *(ii)* HGNN layers, and *(iii)* an MLP decoder.

Communication Hypergraph. We detail several strategies to generate a hypergraph $\mathcal{H}$ in Section 3.2. We include hyperedge features ω_{je} for each $e \in \mathcal{H}$ and $j \in T(e)$. This is set to be a three-dimensional vector, consisting of the relative positional co-ordinates and the Manhattan distance between the agent and the hyperedge centre, which is taken to be the position of the head agent or, in the general case, the centroid of the head agents.

Hypergraph Attention Network. We design our HGNN in a message-passing manner, with messages going from the tail nodes to the hyperedges to the head nodes. Concretely, let $\mathbf{x}_i^{(l)}$ denote the node feature of agent i at layer l. The layer update proceeds with:

$$\mathbf{x}_i^{(l+1)} = \sigma\left(\mathbf{W}_R^{(l)}\,\mathbf{x}_i^{(l)} + \sum_{e\in\Gamma(i)\wedge i\in H(e)} \alpha_{ie}^{(l)}\,\left(\mathbf{W}_h^{(l)}\,\mathbf{h}_e^{(l)}\right)\right) \tag{1}$$

$$\alpha_{ie}^{(l)} = \mathrm{softmax}\left[\mathrm{LeakyReLU}\left(\left(\mathbf{x}_i^{(l)}\right)^\top\left(\Theta_h^{(l)}\,\mathbf{h}_e^{(l)}\right)\right)\right] \tag{2}$$

$$\mathbf{h}_e^{(l)} = \sum_{j\in T(e)} \alpha_{ej}^{(l)}\,\left(\mathbf{W}_n^{(l)}\,\mathbf{x}_j^{(l)} + \mathbf{W}_e^{(l)}\,\mathbf{w}_{je}\right) \qquad \text{(hyperedge representation)} \tag{3}$$

$$\alpha_{ej}^{(l)} = \mathrm{softmax}\left[\mathrm{LeakyReLU}\left(\left(\sum_{i\in H(e)} \mathbf{x}_i^{(l)}/|H(e)|\right)^\top\left(\Theta_n^{(l)}\,\mathbf{x}_j^{(l)} + \Theta_e^{(l)}\,\mathbf{w}_{je}\right)\right)\right] \tag{4}$$

Here, $\mathbf{w}_{je} = \phi(\omega_{je})$ is the hyperedge feature processed via an MLP ϕ, $\alpha_{ej}^{(l)}$ and $\alpha_{ie}^{(l)}$ are the normalised attention weights, $\mathbf{W}_{\{R,n,e,h\}}^{(l)}$ and $\Theta_{\{n,e,h\}}^{(l)}$ are the learnable weights, and σ is a non-linearity. The above design reflects the necessity of modelling variable hypergraph and hyperedge sizes. Instead of architectures with fixed and uniquely identifiable relation types (Busbridge et al., 2019), `HMAGAT` is derivative from attentional HGNNs (Bai et al., 2021; Chen et al., 2020), which can accommodate dynamic and flexible hypergraph structures.

3.2 Hypergraph Generation

Unlike GNNs, which naturally introduce pairwise interactions over a communication graph, defining hypergraphs for MAPF that capture group interactions is not trivial. We focus on directed hypergraphs with singleton heads and multi-node tails. An intuitive approach is to assign a colour—potentially multiple colours—to each vertex $v \in V$, based on geometric adjacency, under the belief that agents in the same local region are more likely to interact collectively. We then form groups according to agent locations and their colours; that is, agents sharing the same colour form the tail of a hyperedge. Concretely, let C be the set of colours, and let $R : V \mapsto 2^{|C|}$ denote the colouring function. Given C and R, Alg. 1 creates a directed hypergraph subject to the constraint of a communication radius R^{comm}. We next discuss strategies to build the colouring R.

Algorithm 1 ColouringBasedHypergraphs

input: colours C, colouring R, agents A, positions $\mathcal{Q}$
1: $\mathcal{E} \leftarrow \emptyset$
2: **for** $v \in A, c \in C$ **do**
3: $\quad T \leftarrow \{u \in A \mid \|\mathcal{Q}[u] - \mathcal{Q}[v]\| \leq R^{\text{comm}} \wedge (\mathcal{Q}[u], c) \in R\}$
4: $\quad$ **if** $T \neq \emptyset$ **then** $\mathcal{E} \leftarrow \mathcal{E} \cup \{(T \cup \{v\}, \{v\})\}$
5: **return** $(A, \mathcal{E})$ ▷ *directed hypergraph*

Lloyd Hypergraphs. A principled approach to workspace partitioning is to employ a Voronoi diagram, which divides the space into regions based on distances to a set of k initial seed locations. By applying Lloyd's algorithm (Lloyd, 1982; Zaman et al., 2024), one can obtain a balanced Voronoi partition with approximately equal-sized regions, regardless of the choice of initial seeds. In principle, each region can be assigned a distinct colour. However, a strict partitioning is not ideal for our setting, since agents may interact across these artificial boundaries. This motivates us to introduce "soft" borders by allowing multiple colours per vertex. Concretely, after the Lloyd's algorithm, we discard half of the least populous colours and reassign the corresponding regions to the colours of their neighbouring regions. An illustration of this process is shown in Figure 3, with the pseudocode provided in Appendix A.

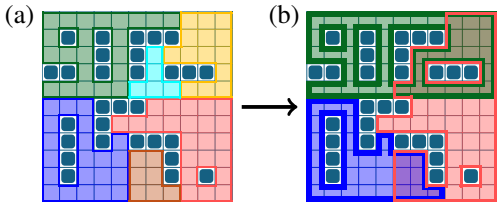

Figure 3: Going from (a) strict partitioning to (b) overlapping groups.

k-means Hypergraphs. Since Lloyd's algorithm requires $O(|V|^3)$ time for updating centroids, it becomes impractical for large graphs. In a faster method, we first *diffuse* colours from k randomly sampled vertices, where each vertex is associated with a k-dimensional vector and each dimension corresponds to one colour. Over a fixed number of iterations T, each vertex updates its vector to the mean of its neighbours' vectors. We then apply k-means clustering (MacQueen, 1967) to these vectors, followed by the "soft boundary operation" described above. This procedure reduces the complexity to $O(k|V|)$. See Appendix A for the detailed procedure.

Shortest Distance-based Hypergraphs. We also explore non-colouring-based hypergraphs. Given an agent i, we say that two agents j and k should be in the tail of a hyperedge with head i if agent i can encounter one agent while visiting the other. We formalise this using the shortest path distances between the agents. We include further details of this strategy in Appendix A.

3.3 Training Pipeline

Model Training. We use the POGEMA toolkit (Skrynnik et al., 2025), also employed in the development of MAPF-GPT (Andreychuk et al., 2025a). Following MAPF-GPT's training protocol, we generate a total of 21K instances, of which, 20% feature randomly placed obstacles, while the remaining 80% are maze-like environments. Map sizes range from 17×17 to 21×21, with 16, 24, or 32 agents. For reference, MAPF-GPT uses 3.75M instances for training. In line with MAPF-GPT, we generate expert trajectories using lacam3 (Okumura, 2024), a state-of-the-art anytime MAPF solver. We adopt a staged timeout strategy with time limits of $[1, 5, 15, 60]$s, where a longer timeout is used only if the shorter ones fail. As the training set does not include challenging situations, most instances were solved within 1 s. HMAGAT, with $R^{\text{comm}} = 7$ and $R^{\text{obs}} = 5$, is trained on the collected expert trajectories using cross-entropy loss. We train for 200 epochs, requiring about 100 hours on an NVIDIA L40S GPU, using the AdamW optimiser (Loshchilov & Hutter, 2019).

Online Expert. To mitigate the distributional shift common in imitation learning, we further apply on-demand dataset aggregation (Ross et al., 2011), collecting new trajectories where the model fails, following MAGAT (Li et al., 2021). To improve the quality of trajectories, after achieving 80% success rate, we check if the length of the model's solution exceeds $\delta_{\text{buf}}\times$ the expert trajectory length, with $\delta_{\text{buf}} = 1.2$. If so, we follow Andreychuk et al. (2025b) and extract instances at every h-th step, where $h = 16$. We then solve these instances using lacam3 with timeouts of $[1, 2, 10]$s, adding trajectories that are $\delta_{\text{buf}}\times$ shorter than the model's trajectory. To keep the training time manageable, we limit the total number of quality improvement expert-calls to 30 per online expert phase.

Post-Training. We follow MAPF-GPT-DDG (Andreychuk et al., 2025b) and apply a post-training phase to further improve the solution quality. This follows the same training procedure as before, but with the quality improvement online expert being called for all 500 instances and with the training consisting of $1 : 3$ ratio of the quality improvement instances and the pre-collected instances, respectively. We run the online expert after every epoch, due to the high accuracy of the model. We run this post-training for 20 epochs, requiring about 30 hours on an NVIDIA L40S GPU.

Temperature Sampling. Previous works have shown that GNNs can achieve high accuracy, but tend to be miscalibrated, having low confidence in their predictions (Hsu et al., 2022; Wang et al., 2021). This is particularly problematic in MAPF, where we sample the next action based on the model's output distribution. Similar problems are likely to exist for HGNNs as well. To tackle this, we use a softmax temperature τ between 0.5 and 1.0. We train an RL module to dynamically adjust τ for each agent based on its local observability and action log-odds. This module is trained for 50 epochs, taking about 3 hours on an NVIDIA L40S GPU. We include further details in Appendix A.

4 Evaluation

Baselines. We compare HMAGAT with the following three state-of-the-art learning-based MAPF solvers: *(i)* MAGAT (Li et al., 2021), an attentional GNN-based model. *(ii)* MAPF-GPT (Andreychuk et al., 2025a), the most capable IL policy to date, using a GPT-like architecture. The authors provided three model sizes: 2M, 6M, and 85M. *(iii)* MAPF-GPT-DDG (Andreychuk et al., 2025b), a fine-tuned version of MAPF-GPT (2M) model, which the authors claim matches the 85M model. We use PIBT-based collision shielding (Okumura et al., 2022) for all these methods, as recommended by Veerapaneni et al. (2025). We also run our evaluation on SSIL (Veerapaneni et al., 2025), another GNN-based IL model, and SCRIMP (Wang et al., 2023) and DCC (Ma et al., 2021), two RL models. However, due to the vast difference in performance, we limit their results to Appendix C.

Metrics. We present the success rates, the average relative sum-of-costs (Rel. SoC) with respect to lacam3 with a 30 s budget and the average runtime per-map. The success rate measures the percentage of instances in which all the agents reach their respective goals.

Evaluation Maps. The commonly used MAPF-benchmarking maps can be divided into seven main categories (Stern et al., 2019): *(i)* `Maze`, *(ii)* `Warehouse`, *(iii)* `Room`, *(iv)* `Game`, *(v)* `City`, *(vi)* `Random` and *(vii)* `Open`.

In our evaluation below, we use *(i)* `Sparse Maze` and *(ii)* `Empty Room` with 256 step limit and 60 s time limit, *(iii)* `Dense Maze`, *(iv)* `Dense Room` and *(v)* `Dense Warehouse` with 512 step limit and 60 s time limit, and *(vi)* `ost003d` with 1024 step limit and 90 s time limit. Since `Random` and `Open` maps are just simpler versions of `Maze` maps, we omit their results. `City` maps are large and sparse, with simple obstacle structures, so we limit their evaluation to Appendix C.

4.1 RESULTS

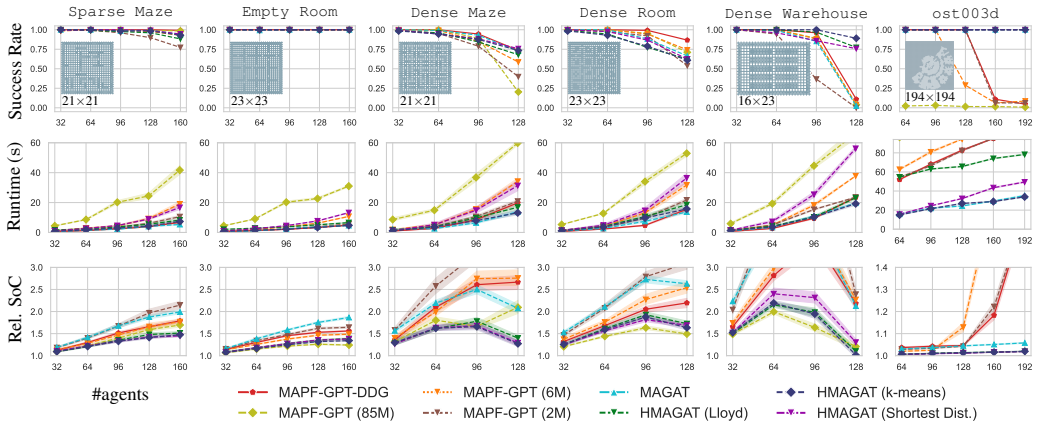

Figure 4: Evaluation of the learning-based MAPF policies, averaged over 128 instances. Transparent regions represent 95% confidence intervals. SoC is reported relative to `lacam3` (w/ 30 s time limit).

Figure 4 shows the results on the different maps. We see that all `HMAGAT` strategies consistently obtain higher quality solutions than `MAGAT`, as well as larger models, like `MAPF-GPT` (2M), `MAPF-GPT` (6M) and `MAPF-GPT-DDG` (2M). `HMAGAT` remains competitive with the largest `MAPF-GPT` (85M) model, while being significantly faster. k-means and Lloyd's `HMAGAT` have consistently high solution quality, with the k-means strategy being faster, especially on the larger maps, as expected. For the rest of the analysis, we focus on k-means hypergraphs.

`HMAGAT` outperforms `MAPF-GPT` (85M) in the `Maze` maps in terms of solution quality, but `MAPF-GPT` (85M) outperforms `HMAGAT` in the `Room` and `Warehouse` maps, except for the highest agent density case for `Dense Warehouse` map. In fact, `HMAGAT` achieves 75+% success rate even in this challenging scenario, while the other methods achieve $<11\%$ success rates. `MAPF-GPT` (85M) is fails to scale to `ost003d`, while `HMAGAT` obtains solutions close to `lacam3`'s quality.

Ablation Study. Figure 5 shows the results of our ablation study over the highest agent density scenarios on each of the small maps, where we start from `MAGAT` and incrementally add components to reach `HMAGAT`. For ease, we fix the hypergraph generation strategy to be k-means colouring-based. The addition of each module improves both the success rate and the solution quality, except for the RL-based temperature sampling, which trades off the success rate for improved solution quality. This shows the importance of each component for the success of `HMAGAT`.

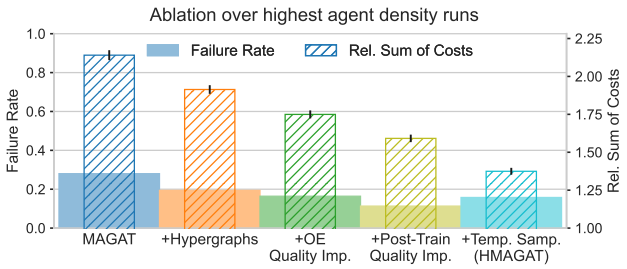

Figure 5: Ablation study. We start from `MAGAT` and incrementally add components to reach `HMAGAT`. We plot the failure rate and Rel. SoC (both the lower the better), over the highest agent density scenarios of the small maps.

Table 1: Comparison of HGNN and GNN, corresponding to `HMAGAT` w/o embellishments and `MAGAT`, on different maps with the maximum agent density. Rel. SoC is relative to `lacam3`.

Metric	Method	Dense Maze	Dense Warehouse	Dense Room	ost003d	Paris
Success Rate	GNN	72.7%	2.3%	66.4%	100.0%	100.0%
	HGNN	75.8%	39.8%	75.0%	100.0%	100.0%
Rel. SoC (95% CI)	GNN	2.07 ± 0.09	2.12 ± 0.10	2.63 ± 0.07	1.06 ± 0.00	1.03 ± 0.00
	HGNN	1.79 ± 0.09	1.78 ± 0.08	2.35 ± 0.06	1.05 ± 0.00	1.02 ± 0.00

4.2 HGNN vs GNN – Deeper Analysis

In this section, we further analyse the difference due to the use of HGNNs over GNNs. To isolate the effects and perform a direct comparison, we focus on a stripped-down version of `HMAGAT`, consisting solely of the replacement of the GNN layers with HGNNs. To emphasise the difference from `HMAGAT`, we refer to this stripped-down version as simply the HGNN model and `MAGAT` as the GNN model, for this analysis.

Table 1 shows the comparison between the two models over different maps. We see that the HGNN model consistently outperforms the GNN model across all maps, especially in the denser maps. This aligns with our hypothesis that HGNNs are better able to capture complex group interactions. The HGNN model also has a higher success rate in the dense scenarios. This gap is particularly pronounced on the `Dense Warehouse` map, where the HGNN model has a success rate of 39.8% compared to the GNN model's 2.3%.

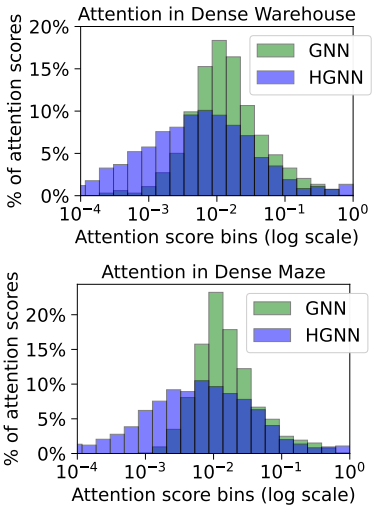

Figure 6: Attention score distribution in the first layer of the models with 128 agent instances.

Attentional Analysis. To better understand the difference in performance of the two models, we analyse the attention scores in the first layer of both the models. Figure 6 shows the distribution of these attention scores over a sample instance of `Dense Warehouse` and `Dense Maze` maps. The GNN model has a high concentration of attention scores in the higher-middle range, which leads to a dilution of attention, resulting in very few agents receiving high attention. On the other hand, the HGNN model has fewer agents in this range, allowing it to maintain more agents in the highest attention score bin. This supports our hypothesis that GNNs struggle in high density scenarios due to the dilution of attention scores, which our HGNN model is able to overcome.

4.3 Hand-Crafted Scenarios

We now analyse the two models on hand-crafted scenarios and showcase the limitations of GNNs due to *(i)* attention dilution in dense scenarios and *(ii)* the inability to effectively capture group interactions, which HGNNs are able to overcome since they do not rely on pairwise interactions.

Scenario 1. Since, in GNNs, the attention scores are computed in a pairwise manner, a high density of agents in unimportant regions will lead to a dilution of attention. To illustrate this, we consider the scenario shown in Figure 7 (left). We can create a sequence of scenarios by starting with 4 agents (agents 0-3 in groups 0-3, respectively) and sequentially adding agents 4-7 to group 3. We focus on the first layer attention scores wrt. agent 0. By design, agent 0 primarily needs to attend to agent 2 and, potentially, agent 1, as their respective greedy paths have intersections with agent 0's. On the other hand, the agents in group 3 are not expected to interfere with agent 0's path, and thus, increasing the number of agents in group 3 should not affect the attention distribution for agent 0.

Table 2 shows the coefficient of variation (CV) of the attention scores. The GNN model exhibits a high variation in the attention scores as we increase the number of agents in less important regions, leading to a dilution of the attention scores in the important regions. In contrast, the HGNN model

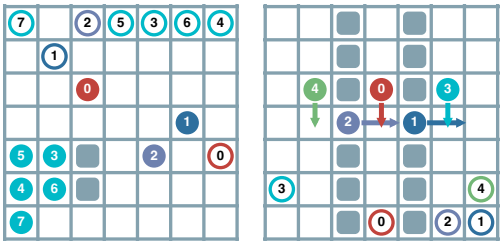

Figure 7: Hand-crafted scenarios. The filled circles represent the agent locations, while the empty ones represent their targets. **Scenario 1** (left): We form 4 groups of agents, shown by different colours here. **Scenario 2** (right): We show the next greedy actions for the agents.

Table 2: CV of attention scores (in the first layer) for agent 0 as we vary the number of agents from 4 to 8 (inclusive both) in the map described in Figure 7 (left).

Method	Grp. 0	Grp. 1	Grp. 2	Grp. 3
GNN	29.4%	20.0%	14.9%	39.2%
HGNN	6.0%	4.3%	7.4%	14.7%
Ratio	4.94	4.66	2.01	2.66

Table 3: Percentage Shapley values for the agents with respect to agent 0 in Figure 7 (right).

Method	Agt. 1	Agt. 2	Agt. 3	Agt. 4	Agts. 3 vs. 4
GNN	15.0%	60.3%	13.1%	11.6%	+13%
HGNN	19.0%	59.6%	19.3%	2.1%	+802%

exhibits a much lower variation, remaining relatively unaffected by the number of agents in group 3, as desired. We include more details on the attention scores in Appendix D.

Scenario 2. In this scenario, we aim to illustrate the ability of HGNNs to capture group interactions as opposed to GNNs. Towards this end, we design the scenario shown in Figure 7 (right), analysing how agents 1-4 influence agent 0. To quantify the influence, we make use of Shapley values (Shapley, 1951; Lundberg & Lee, 2017), taking the mean of the absolute Shapley values for each plausible action-class log-odds prediction.

By design, we expect the following influence ranking: Agent 2 $>$ Agent 3 $\approx$ Agent 1 $>$ Agent 4. This is because agent 2's next greedy action conflicts with agent 0's. Agent 3's greedy path will want agent 1 to occupy this conflicting cell, leading to a group interaction effect and thus, a similar high influence on agent 0's actions. Since agent 2 is already moving away from agent 4, agent 4 is expected to have a low influence. From a purely pairwise perspective, agents 3 and 4 have mirrored start and goal locations wrt. agent 0, and thus, should have similar influence on agent 0's actions. However, due to group interactions, we expect agent 3 to have a higher influence than agent 4.

Table 3 shows the percentage Shapley values for the GNN and HGNN models. As expected, we see that both models assign the highest influence to agent 2. For the GNN model, agents 1, 3 and 4 have a similar influence, with agent 3 having only 13% more influence than agent 4. This highlights the inability of the GNN model to capture group interactions effectively. On the other hand, in the HGNN model, agent 3 has a significantly higher influence (802% more) than agent 4, showcasing the ability of HGNNs to capture group interactions and distinguish between the agents.

5 Conclusion

In this work, we explored MAPF as a representative multi-agent problem with complex interactions. We proposed `HMAGAT`, an MAPF solver based on hypergraph neural networks (HGNNs), as a solution to model higher order interactions. We empirically demonstrated that our approach outperforms current state-of-the-art methods, which only consider pairwise interactions. `HMAGAT` achieves superior performance to `MAPF-GPT` (85M), a 85$\times$ larger model, trained on 100$\times$ more data. We perform further analyses to show that, indeed, the HGNNs help to capture group interactions better than GNNs via explicit group modelling. Our results show how higher order representational learning models can lead to improved model performance, while lowering the sample complexity and model size, strongly motivating future research to explore hypergraph-based methods for challenging multi-agent problems with highly coupled interactions. As our model achieves SoTA performance with a significantly smaller parameter count than previous works, our results demonstrate that better inductive biases, such as hypergraph-based interaction modelling, are a complementary strategy to training larger models when tackling difficult multi-agent problems.

REFERENCES

Jeffrey L Adler and Victor J Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5):433–454, 2002. ISSN 0968-090X. doi: https://doi.org/10.1016/S0968-090X(02)00030-X. URL https://www.sciencedirect.com/science/article/pii/S0968090X0200030X.

Ameya Agaskar, Sriram Siva, William Pickering, Kyle O'Brien, Charles Kekeh, Ang Li, Brianna Gallo Sarker, Alicia Chua, Mayur Nemade, Charun Thattai, et al. Deepfleet: Multi-agent foundation models for mobile robots. *arXiv preprint arXiv:2508.08574*, 2025.

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. Mapf-gpt: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2025a.

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. Advancing learnable multi-agent pathfinding solvers with active fine-tuning. *arXiv preprint arXiv:2506.23793*, 2025b.

Song Bai, Feihu Zhang, and Philip H.S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107637. URL https://www.sciencedirect.com/science/article/pii/S0031320320304404.

Guillermo Bernárdez, José Suárez-Varela, Albert López, Xiang Shi, Shihan Xiao, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Magnneto: A graph neural network-based multi-agent system for traffic engineering. *IEEE Transactions on Cognitive Communications and Networking*, 9(2):494–506, 2023. doi: 10.1109/TCCN.2023.3235719.

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

Chaofan Chen, Zelei Cheng, Zuotian Li, and Manyi Wang. Hypergraph attention networks. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1560–1565. IEEE, 2020.

Tzvika Geft and Dan Halperin. Refined hardness of distance-optimal multi-agent path finding. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 481–488, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? 2022.

Jiewu Leng, Weinan Sha, Zisheng Lin, Jianbo Jing, Qiang Liu, and Xin Chen. Blockchained smart contract pyramid-driven multi-agent autonomous process control for resilient individualised manufacturing towards industry 5.0. *International Journal of Production Research*, 61(13):4302–4321, 2023. doi: 10.1080/00207543.2022.2089929. URL https://doi.org/10.1080/00207543.2022.2089929.

Jiachen Li, Chuanbo Hua, Hengbo Ma, Jinkyoo Park, Victoria Dax, and Mykel J Kochenderfer. Multi-agent dynamic relational reasoning for social robot navigation. *arXiv preprint arXiv:2401.12275*, 2024a.

Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 11785–11792. IEEE, 2020.

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 (RA-L)*, 2021.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024b.

Weilong Lin, Xinhua Zeng, Chengxin Pang, Jing Teng, and Jing Liu. Dyhgdat: Dynamic hypergraph dual attention network for multi-agent trajectory prediction. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16662–16668. IEEE, 2024.

Ziheng Liu, Jiayi Zhang, Enyu Shi, Zhilong Liu, Dusit Niyato, Bo Ai, and Xuemin Shen. Graph neural network meets multi-agent reinforcement learning: Fundamentals, applications, and future directions. *IEEE Wireless Communications*, 31(6):39–47, 2024.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. doi: 10.1109/TIT.1982.1056489.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning and Representation (ICLR)*, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman (eds.), *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297. University of California Press, 1967.

Keisuke Okumura. Engineering lacam*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, 2024.

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

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

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 (RA-L)*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Lloyd S. Shapley. *Notes on the N-Person Game — II: The Value of an N-Person Game*. RAND Corporation, Santa Monica, CA, 1951. doi: 10.7249/RM0670.

Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin Yakovlev, and Aleksandr Panov. Pogema: A benchmark platform for cooperative multi-agent pathfinding. In *Proceedings of the International Conference on Learning and Representation (ICLR)*, 2025.

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

Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012. doi: 10.1109/TCIAIG.2012.2197681.

Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pp. 1261–1263, 2010.

Yimin Tang, Xiao Xiong, Jingyi Xi, Jiaoyang Li, Erdem Bıyık, and Sven Koenig. Railgun: A unified convolutional policy for multi-agent path finding across different environments and tasks. *arXiv preprint arXiv:2503.02992*, 2025.

Rishi Veerapaneni, Qian Wang, Kevin Ren, Arthur Jakobsson, Jiaoyang Li, and Maxim Likhachev. Improving learnt local mapf policies with heuristic search. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2024.

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. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020. doi: 10.1109/LRA.2020.3026638.

Weizheng Wang, Aniket Bera, and Byung-Cheol Min. Hyper-samarl: Hypergraph-based coordinated task allocation and socially-aware navigation for multi-robot systems. *arXiv preprint arXiv:2409.11561*, 2024.

Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. 2021.

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. IEEE, 2023.

Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9, Mar. 2008. doi: 10.1609/aimag.v29i1.2082. URL `https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2082`.

Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6498–6507, 2022.

Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Generalizing lloyd's algorithm for graph clustering. *SIAM Journal on Scientific Computing*, 46 (5):A2819–A2847, 2024. doi: 10.1137/23M1556800. URL `https://doi.org/10.1137/23M1556800`.

Delin Zhao, Yanbo Shan, Chang Liu, Shenghang Lin, Yingxin Shou, and Bin Xu. Sdhn: Skewness-driven hypergraph networks for enhanced localized multi-robot coordination. *arXiv preprint arXiv:2504.06684*, 2025.

Tianyu Zhu, Xinli Shi, Xiangping Xu, Jie Gui, and Jinde Cao. Hypercomm: Hypergraph-based communication in multi-agent reinforcement learning. *Neural networks: the official journal of the International Neural Network Society*, 178:106432, 2024.

A Further Details for HMAGAT

A.1 Temperature Sampling Module

We consider a lightweight temperature sampling module to predict the softmax temperature τ. To train this module, we consider an actor-critic setup. For each agent, the actor takes the log-odds predicted by the model, the number of agents in its FOV, the number of obstacles in its FOV, and the distance and relative positional vector to its goal as input, passing them through a 2-layer MLP with ReLU activations, outputting a scalar $\tau \in [0.5, 1.0]$ for each agent, by applying a sigmoid activation and scaling the output. The critic takes the same input, passing them through a 2-layer

Algorithm 2 SHORTESTDISTANCEBASEDHYPERGRAPHS

input: grid $G = (V, E)$, agents A, agent positions $\mathcal{Q}$
params: communication radius R^{comm}, dynamic slack variable ϵ

$\mathcal{E} \leftarrow \emptyset$ ▷ *set of hyperedges*
for $v \in A$ **do**
  $H \leftarrow \{v\}$ ▷ *head of hyperedge*
  $A_C \leftarrow \{u \in A \mid \|\mathcal{Q}[u] - \mathcal{Q}[v]\| \leq R^{\text{comm}}\} \setminus \{v\}$ ▷ *communicating agents*
  $E_C \leftarrow \emptyset$ ▷ *initialize candidate edges*
  for $u \in A_C$ **do**
    for $w \in A_C \setminus \{u\}$ **do**
      if $\mathsf{dist}(\mathcal{Q}[v], \mathcal{Q}[u]) + \mathsf{dist}(\mathcal{Q}[u], \mathcal{Q}[w]) \leq \max(\mathsf{dist}(\mathcal{Q}[v], \mathcal{Q}[u]), \mathsf{dist}(\mathcal{Q}[v], \mathcal{Q}[w])) + \epsilon$ **then**
        $E_C \leftarrow E_C \cup \{(u, w), (w, u)\}$
  $C \leftarrow$ GETCLIQUES(A_C, E_C)
  for $T \in C$ **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(T \cup \{v\}, H)\}$
return $\mathcal{E}$

Algorithm 3 GRIDCOLOURING

input: grid $G = (V, E)$
params: number of initial colours k', number of final colours k, number of iterations n

$R', C' \leftarrow$ LLOYDSONGRAPH(G, k', n) ▷ *Rigid colouring from Lloyd's*
$C \leftarrow k$-most populous colours in R' ▷ *We use $k = k'/2$*
$R \leftarrow \{(v, c) \in R' \mid c \in C\}$ ▷ *Discard least populous colourings*
$U \leftarrow V \setminus \{v \mid \exists c \in C.(v, c) \in R\}$ ▷ *Initial uncoloured vertices*
while $\exists v \in V \,.\, !\exists c \in C \,.\, (v, c) \in R$ **do** ▷ *While there are uncoloured vertices*
  $R_{\text{old}} \leftarrow R$
  for $u \in U$ **do**
    $R \leftarrow R \cup \{(u, c) \in V \times C \mid \exists v \in \mathsf{neigh}(u) \,.\, (v, c) \in R_{\text{old}}\}$ ▷ *Take neighbours' colours*
return R, C ▷ *Less rigid colouring*

MLP with ReLU and outputs a scalar value for the instance by averaging the outputs for all agents. We consider a simple reward function, where we give a reward of $+1$ to each agent if all agents reach their goals, else a reward of -1, in order to encourage higher success rates, while also encouraging shorter paths.

We train this module using the PPO algorithm (Schulman et al., 2017), with a clip ratio of 0.2, a discount factor of 0.99, and a GAE λ of 0.95. We use a learning rate of 3×10^{-4} and a batch size of 64. We train this module for 50 epochs, requiring about 3 hours on an NVIDIA L40S GPU. These are trained on 85 instances, each with 32 agents, on a $20 + 80\%$ combination of `Random` and `Maze` maps, with 15 separate instances for validation.

A.2 HYPERGRAPH GENERATION

Colouring-based Hypergraphs. Algorithm 3 shows the pseudocode for generating a colouring for a given grid using Lloyd's algorithm, with overlapping regions. Algorithm 4 shows the pseudocode for generating a colouring using k-means clustering, with overlapping regions.

Lloyds versus k-means. Lloyd's algorithm for graphs employs Floyd-Warshall's algorithm to update centroids, which has a time complexity of $O(\sum_{c \in C} |s_c|^3)$, where s_c is the size of cluster c. The cluster sizes can be imbalanced, leading to a worst-case time complexity of $O(|V|^3)$. In contrast, k-means clustering has a time complexity of $O(\sum_{c \in C} d \times |s_c|)$ for the updates, because it only requires computing the mean of the features of the vertices in the cluster, where d is the feature dimension. Since, in our case, $d = k$ (the number of clusters) and $\sum_{c \in C} |s_c| = |V|$, this complexity becomes $O(k \times |V|)$.

Shortest Distance-based Hypergraphs. Algorithm 2 shows the pseudocode for generating shortest distance-based hypergraphs. In order to get faster clique computations, we simply sample cliques by randomly selecting an agent from the communicating agents, and then greedily adding agents to the clique by checking if the new agent is connected to all the existing agents in the clique.

Algorithm 4 KMEANSCOLOURING

input: grid $G = (V, E)$
params: number of initial colours k', number of final colours k, number of iterations n

$X \leftarrow \mathbf{0}_{|V| \times k'}$ ▷ *Initial colour vectors*
$X \leftarrow$ Randomly assign k' vertices to unique colours (i.e. set $X[v, c] := 1$) ▷ *Random seeding*
for $i \in [1, n]$ **do** ▷ *Diffuse colours*
  $X_{\text{old}} \leftarrow X$
  for $v \in V$ **do**
    $X[v] \leftarrow \sum_{u \in \mathsf{neigh}(v)} X_{\text{old}}[u]$
    $X[v] \leftarrow X[v] / \|X[v]\|_1$ ▷ *Mean of neighbours' colours*
$R', C' \leftarrow$ KMEANS(X, k', n) ▷ *Rigid colouring from k-means*
$C \leftarrow k$-most populous colours in R' ▷ *We use $k = k'/2$*
$R \leftarrow \{(v, c) \in R' \mid c \in C\}$ ▷ *Discard least populous colourings*
$U \leftarrow V \setminus \{v \mid \exists c \in C.(v, c) \in R\}$ ▷ *Initial uncoloured vertices*
while $\exists v \in V \,.\, !\exists c \in C \,.\, (v, c) \in R$ **do** ▷ *While there are uncoloured vertices*
  $R_{\text{old}} \leftarrow R$
  for $u \in U$ **do**
    $R \leftarrow R \cup \{(u, c) \in V \times C \mid \exists v \in \mathsf{neigh}(u) \,.\, (v, c) \in R_{\text{old}}\}$ ▷ *Take neighbours' colours*
return R, C ▷ *Less rigid colouring*

Table 4: Map details.

Map	Size (H × W)	Obstacle Density	Step Limit	Time Limit
Sparse Maze	21×21	$8 - 24\%$	256	60 s
Empty Room	23×23	~20%	256	60 s
Sparse Warehouse	22×23	~32%	256	60 s
Dense Maze	21×21	$30 - 40\%$	512	60 s
Dense Room	23×23	$34 - 42\%$	512	60 s
Dense Warehouse	16×23	~43%	512	60 s
ost003d	194×194	~65%	1024	90 s
lak303d	194×194	~61%	1024	90 s
Paris	256×256	~28%	1024	90 s
Berlin	256×256	~27%	1024	90 s

We repeat this process until we have each communicating agent in at least one clique, each time selecting one of the non-selected agents as the starting agent. We repeat the process if we have less than 5 total cliques, in order to ensure that we have a reasonable number of sampled cliques even in highly connected distance graphs. To lower the computation time, we only generate new hypergraphs every 5 timesteps.

B RADAR PLOT DETAILS

Figure 1c shows the performance of various learning-based MAPF solvers on the small maps, i.e. on Sparse Maze, Empty Room, Sparse Warehouse, Dense Maze, Dense Room and Dense Warehouse. More details on the maps can be found in Appendix C. Figure 8 shows more detailed results on these maps. We define the metrics used in the radar plot as follows:

$$\text{Sol. Quality} = \begin{cases} \text{SoC}_{\text{best}}/\text{SoC} \\ 0 \end{cases}, \text{if no solution found} \tag{5}$$

$$\text{Scalability} = \begin{cases} \frac{\text{runtime(\#agents)}}{\text{\#agents}} \cdot \frac{\text{Min. \#agents}}{\text{runtime(Min. \#agents)}} \\ 0 \end{cases}, \text{if timed out} \tag{6}$$

C Further Evaluation Results

C.1 More Evaluation Maps

We provide further evaluation results on more maps with more methods here. Table 4 shows the details for each map. `ost003d` and `lak303d` are `Game` maps, while `Paris` and `Berlin` are `City` maps, all taken from the MovingAI pathfinding repository (Sturtevant, 2012).

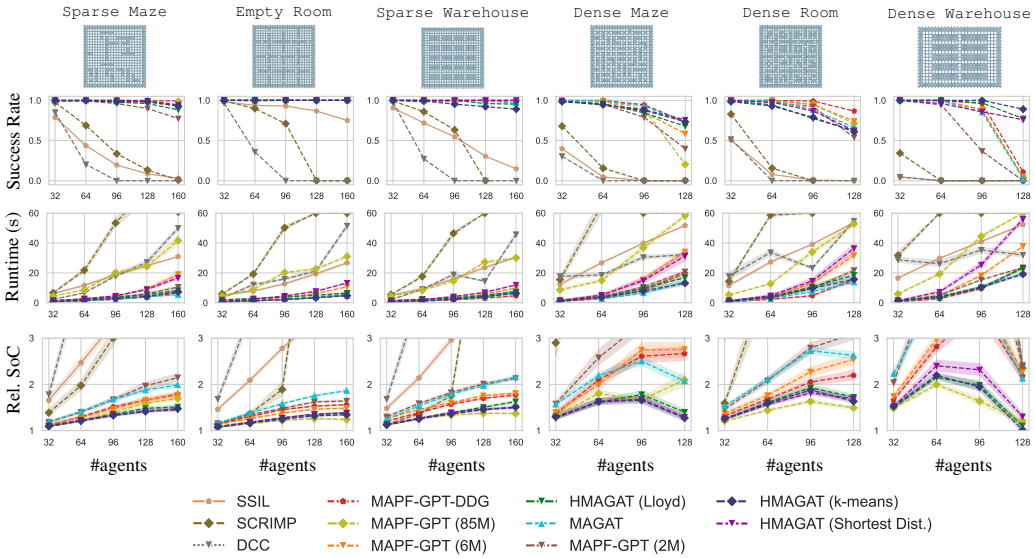

Figure 8: Evaluation of the learning-based MAPF policies on the small maps, averaged over 128 instances. Transparent regions represent 95% confidence intervals. SoC is reported relative to `lacam3` (w/ 30 s time limit).

C.2 Comparison of HMAGAT Variants

Small maps. Figure 8 shows the results on the small maps. On the sparse maps, all `HMAGAT` variants perform nearly identically in terms of SoC. They also have similar success rates, except for k-means-based `HMAGAT` on `Sparse Warehouse`, where it has slightly worse success rate. k-means-based and Lloyds-based `HMAGAT` have similar runtimes, while the shortest-distance based `HMAGAT` is slightly slower, with the gap being more pronounced the more agents there are. This is expected because unlike the colouring-based methods that compute the colouring irrespective of the agent configurations, the shortest-distance based method's time complexity increases with the number of agents.

On the `Dense Maze` and `Dense Room` maps, all `HMAGAT` variants again have similar SoC and success rates. However, on `Dense Warehouse`, the shortest-distance based `HMAGAT` has a worse SoC, and both the shortest-distance based and Lloyds-based `HMAGAT` have slightly worse success rates. In these scenarios, the shortest-distance based `HMAGAT` is again the slowest, with the gap being much more pronounced. This is expected because unlike the colouring-based methods that only need to compute the colouring once, the shortest-distance based method needs to compute the shortest distances every 5 timesteps, which means that for the more complex maps, where the agents take longer to reach their goals, the overhead of computing the shortest distances is more pronounced.

Large maps. Figure 9 shows the results on the large maps. On the `Game` maps, again all `HMAGAT` variants perform nearly identically in terms of SoC and all have perfect success rates. k-means-based `HMAGAT` is the fastest, with the shortest-distance based `HMAGAT` being slightly slower, with the gap increasing with the number of agents. Lloyds-based `HMAGAT` is the slowest, although the gap does not increase with the number of agents. This is expected because the colouring is computed

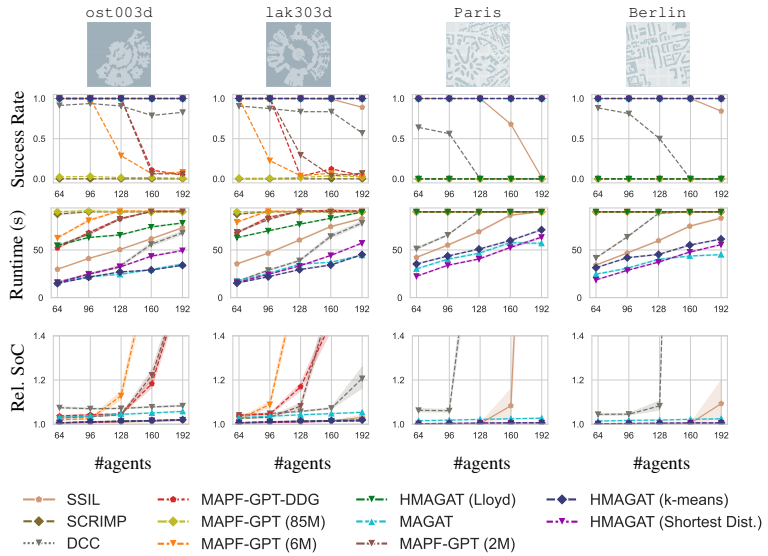

Figure 9: Evaluation of the learning-based MAPF policies on the large maps, averaged over 128 instances. Transparent regions represent 95% confidence intervals. SoC is reported relative to `lacam3` (w/ 30 s time limit).

only once and is independent of the number of agents. Lloyds is slower than k-means as noted in Appendix A and this becomes apparent here, where $|V|$ is large.

On the `City` maps, k-means-based and shortest-distance based `HMAGAT` perform nearly identically in terms of SoC and have perfect success rates. Lloyds-based `HMAGAT`, on the other hand, times out in these scenarios. This is because $|V|$ here is even larger than the `Game` maps. We see that the shortest-distance based `HMAGAT` is faster than k-means-based `HMAGAT` here, although the gap decreases with the number of agents. This is expected because the total time complexity of hypergraph generation in shortest-distance based method depends on the number of time steps and the number of agents, while for the k-means-based method, it depends on $|V|$. Since we are dealing with a large and sparse map, $|V|$ is large leading to a higher time complexity for the k-means-based method, but not for the shortest-distance based method, which is independent of $|V|$ (assuming the number of time steps as an independent variable). However, as the number of agents increases, the time complexity of the shortest-distance based method increases, leading to a decrease in the gap.

We see that k-means-based `HMAGAT` obtains a good balance between performance and speed across all maps, and so, for the rest of the analysis, we focus on this variant.

C.3 Comparison of `HMAGAT` with other methods

Small maps. Figure 8 shows the results on the small maps. `HMAGAT` consistently outperforms `MAGAT`, `MAPF-GPT-DDG`, `MAPF-GPT` (6M), `MAPF-GPT` (2M), `SSIL`, `SCRIMP` and `DCC` in terms of SoC and success rates across all maps and agent counts. The only exception being `MAPF-GPT-DDG` and `MAPF-GPT` on `Dense Room`, where they have higher success rates, but still have worse SoC.

On the sparse maps, `HMAGAT` and `MAPF-GPT` (85M) achieve similar SoC, with `HMAGAT` having slightly worse performance on the highest agent count scenario for `Empty Room` and `Sparse Warehouse`, but better performance on `Sparse Maze`. On the `Dense Maze` map, `HMAGAT` consistently outperforms `MAPF-GPT` (85M), with the gap increasing a fair amount for the highest agent count scenario. For the `Dense Room` and `Dense Warehouse` maps, `MAPF-GPT` (85M) has a slightly better SoC, except for the highest agent count scenario on `Dense Warehouse`, where `HMAGAT` has a better SoC. In this scenario, `HMAGAT` has a significantly higher success rate, achieving 89.1% success rate compared to 4% for `MAPF-GPT` (85M).

Notably, HMAGAT is significantly faster than MAPF-GPT (85M) across all maps and agent counts, and has a lower slope in the runtime vs. number of agents plots.

Large maps. Figure 9 shows the results on the large maps. HMAGAT consistently outperforms MAGAT, MAPF-GPT (85M), MAPF-GPT-DDG, MAPF-GPT (6M), MAPF-GPT (2M), SCRIMP and DCC. HMAGAT and MAGAT are the only two methods that are able to easily scale to these maps and have perfect success rates across all maps and agent counts. SSIL achieves a similar SoC to HMAGAT for the lower agent count scenarios, but seems unable to scale to various higher agent count scenarios.

HMAGAT consistently achieves a Rel. SoC close to 1.0, i.e. it produces solutions close to lacam3's quality.

MAPF-GPT (85M) and MAPF-GPT-DDG. Andreychuk et al. (2025b) claim that MAPF-GPT-DDG matches the performance of MAPF-GPT (85M), and, at times, even outperforms it. However, in our experiments, we see that MAPF-GPT-DDG consistently underperforms MAPF-GPT (85M), except for the large maps, where MAPF-GPT (85M) fails to scale. We believe that this discrepancy is due to the fact that Andreychuk et al. (2025b) evaluate these models without PIBT-based collision shielding (Okumura et al., 2022; Veerapaneni et al., 2024). MAPF-GPT-DDG likely produces solutions with lower collisions, due to the finetuning, leading to a better performance than MAPF-GPT (85M) when not using any advanced collision shielding. To verify this, we evaluated MAPF-GPT-DDG and MAPF-GPT (85M) without PIBT-based collision shielding on Sparse Maze, Empty Room and Sparse Warehouse maps with 128 agents each. Table 5 shows the results. We see that MAPF-GPT (85M) struggles even on these sparse maps and MAPF-GPT-DDG consistently outperforms it, verifying our hypothesis.

However, as seen in Figure 8, when using PIBT-based collision shielding, MAPF-GPT (85M) consistently outperforms MAPF-GPT-DDG.

Table 5: Comparison of MAPF-GPT (85M) and MAPF-GPT-DDG without PIBT-based collision shielding. Averaged over 128 instances, each with 128 agents.

Metric	Method	Sparse Maze	Empty Room	Sparse Warehouse
Success Rate	MAPF-GPT (85M)	6.3%	2.3%	39.1%
	MAPF-GPT-DDG	76.6%	96.9%	99.2%
Rel. SoC (95% CI)	MAPF-GPT (85M)	6.93 ± 0.24	5.81 ± 0.14	5.61 ± 0.23
	MAPF-GPT-DDG	3.81 ± 0.20	3.02 ± 0.14	3.05 ± 0.11

C.4 HGNN vs GNN on Maximum Agent Density

Table 6 shows the performance of HGNN and GNN on different maps with the maximum agent density with additional maps compared to Table 1. We see that HGNN consistently outperforms GNN in terms of solution quality.

D Further Details on Hand Crafted Scenarios

Scenario 1. To calculate $a_{i,j}^{(l)}$ the attention a node i pays to an agent j in the l-th layer of HMAGAT, we simply use the following sum:

$$a_{i,j}^{(l)} = \sum_{e \in \Gamma(i) \wedge i \in H(e) \wedge j \in T(e)} \alpha_{ie}^{(l)} \, \alpha_{ej}^{(l)} \tag{7}$$

We present the attention scores for the GNN and HGNN models in the first layer Tables 7 and 8. We also present the attention scores averaged across all layers in Tables 9 and 10, with the coefficient of variation in Table 11. We see that the HGNN model consistently has a lower coefficient of variation across all groups, indicating that it is more robust to the increase in the number of agents in low-importance regions.

Table 6: Comparison of HGNN and GNN, corresponding to HMAGAT w/o embellishments and MAGAT, on different maps with the maximum agent density. Results are over 128 instances per map. Rel. SoC is the sum-of-costs relative to lacam3, for which we show the 95% confidence intervals.

Metric	**Method**	Sparse Maze	Sparse Warehouse	Sparse Room	ost003d	lak303d
Success Rate	GNN	93.8%	96.9%	100.0%	100.0%	100.0%
	HGNN	93.0%	100.0%	100.0%	100.0%	100.0%
Rel. SoC	GNN	1.99 ± 0.04	2.16 ± 0.04	1.86 ± 0.02	1.06 ± 0.00	1.05 ± 0.00
	HGNN	1.91 ± 0.04	1.95 ± 0.02	1.69 ± 0.02	1.05 ± 0.00	1.04 ± 0.00

Metric	**Method**	Dense Maze	Dense Warehouse	Dense Room	Paris	Berlin
Success Rate	GNN	72.7%	2.3%	66.4%	100.0%	100.0%
	HGNN	75.8%	39.8%	75.0%	100.0%	100.0%
Rel. SoC	GNN	2.07 ± 0.09	2.12 ± 0.10	2.63 ± 0.07	1.03 ± 0.00	1.02 ± 0.00
	HGNN	1.79 ± 0.09	1.78 ± 0.08	2.35 ± 0.06	1.02 ± 0.00	1.02 ± 0.00

Table 7: GNN Attention scores in the first layer for agent 0 over the different groups in Figure 7 (left).

# Agents	Group 0	Group 1	Group 2	Group 3
4 Agents	0.394750	0.257375	0.192023	0.155852
5 Agents	0.286059	0.234413	0.186970	0.292558
6 Agents	0.251166	0.202690	0.161970	0.384175
7 Agents	0.215447	0.172166	0.146812	0.465576
8 Agents	0.194606	0.159428	0.135639	0.510326

Table 8: HGNN Attention scores in the first layer for agent 0 over the different groups in Figure 7 (left).

# Agents	Group 0	Group 1	Group 2	Group 3
4 Agents	0.024621	0.104523	0.493603	0.377252
5 Agents	0.027046	0.110814	0.583928	0.278213
6 Agents	0.024019	0.113196	0.603738	0.259047
7 Agents	0.023478	0.102690	0.571674	0.302157
8 Agents	0.023619	0.104273	0.564101	0.308007

Table 9: GNN Attention scores averaged across layers for agent 0 over the different groups in Figure 7 (left).

# Agents	Group 0	Group 1	Group 2	Group 3
4 Agents	0.170300	0.381573	0.190819	0.257309
5 Agents	0.130953	0.349720	0.188699	0.330629
6 Agents	0.117691	0.329456	0.171886	0.380967
7 Agents	0.109234	0.278605	0.164050	0.448111
8 Agents	0.099965	0.249049	0.156532	0.494454

Table 10: HGNN Attention scores averaged across layers for agent 0 over the different groups in Figure 7 (left).

# Agents	Group 0	Group 1	Group 2	Group 3
4 Agents	0.103088	0.274550	0.305216	0.317146
5 Agents	0.119637	0.247633	0.350739	0.281991
6 Agents	0.118572	0.239537	0.334259	0.307632
7 Agents	0.120345	0.219097	0.329250	0.331308
8 Agents	0.117992	0.221249	0.319412	0.341347

Table 11: Coefficient of variation of the average-attention scores (across layers) when varying the number of agents in group 3 in Figure 7 (left).

Method	Group 0	Group 1	Group 2	Group 3
GNN	21.85%	16.87%	8.63%	24.54%
HGNN	6.24%	9.39%	5.17%	7.26%
GNN / HGNN	3.5012	1.7957	1.6686	3.3794

We have seen that the HGNN model is able to handle the increase in the number of agents in less important regions. However, we also need to ensure that when the number of agents in the important regions increases, we do not see a significant decrease in the attention scores for this region. Towards this end, we construct a scenario as shown in Figure 10.

In contrast to the scenario described in Figure 7 (left), we now have agent 0's greedy path overlapping with paths of agents in group 3, i.e the group for which we vary the number of agents. Thus, agent 0 is expected to primarily attend to agents in group 3. Table 12 shows the HGNN attention scores for this scenario. We see that the attention scores for group 3 stay high as we vary the number of agents, indicating that the HGNN model is able to maintain high attention scores for the important region.

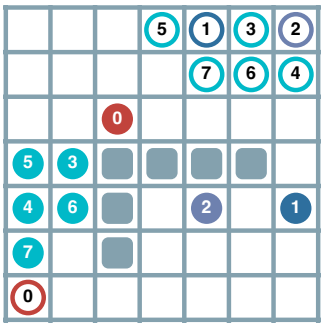

Figure 10: Hand-crafted scenario to illustrate the effect of increasing number of agents in important regions on the attention scores. Filled circles represent the agent locations, while the empty ones represent their targets. Agents are grouped via colours, with groups 0 to 2 consisting of their corresponding agents only, while group 3 consists of agents 3 to 7. We start with the agents numbered from 0 to 3, and add more agents to group 3, in order of their numbering.

Scenario 2. While calculating the Shapley values, we average the scores between the ones computed on the original map shown in Figure 7 (right) and map formed by horizontally flipping it, in order to reduce the effect of any asymmetries learnt by the models.

Table 12: HGNN Attention scores in the first layer over groups.

# Agents	Group 0	Group 1	Group 2	Group 3
4 Agents	0.000949	0.001296	0.001605	0.996150
5 Agents	0.002126	0.000187	0.000252	0.997436
6 Agents	0.002358	0.000235	0.000335	0.997073
7 Agents	0.002095	0.002070	0.002800	0.993035
8 Agents	0.005777	0.003658	0.005720	0.984845