Graph Attention-Guided Search for Dense Multi-Agent Pathfinding

Rishabh Jain[1*], **Keisuke Okumura**[1,2*], **Michael Amir**[1], **Amanda Prorok**[1]

[1]University of Cambridge, UK
[2]National Institute of Advanced Industrial Science and Technology (AIST), Japan
{rj412, ko393, ma2151, asp45}@cst.cam.ac.uk

Abstract

Finding near-optimal solutions for dense multi-agent pathfinding (MAPF) problems in real-time remains challenging even for state-of-the-art planners. To this end, we develop a hybrid framework that integrates a learned heuristic derived from MAGAT, a neural MAPF policy with a graph attention scheme, into a leading search-based algorithm, LaCAM. While prior work has explored learning-guided search in MAPF, such methods have historically underperformed. In contrast, our approach, termed `LaGAT`, outperforms both purely search-based and purely learning-based methods in dense scenarios. This is achieved through an enhanced MAGAT architecture, a pre-train–then–fine-tune strategy on maps of interest, and a deadlock detection scheme to account for imperfect neural guidance. Our results demonstrate that, when carefully designed, hybrid search offers a powerful solution for tightly coupled, challenging multi-agent coordination problems.

Code — https://github.com/proroklab/lagat
Extended version — https://arxiv.org/abs/2510.17382

1 Introduction

Multi-agent pathfinding (MAPF) is a fundamental problem in multi-robot coordination, where the goal is to compute collision-free paths that efficiently guide a team of agents to their respective destinations. Since the early 2010s, driven by applications in warehouse automation, MAPF has been extensively studied in the AI planning community. More recently, its well-defined and discrete problem structure has made MAPF an attractive benchmark for advanced machine learning techniques tailored to multi-agent settings (Prorok et al. 2021; Alkazzi and Okumura 2024). These two streams form the main research thrusts in MAPF today:

- Developing real-time *centralized* planners based on heuristic search, aiming for reliable algorithms that offer scalability with respect to the number of agents and (near-)optimal solution quality;
- Developing robust *decentralized* neural policies that operate under local observations, learn complex agent interactions, and generalize to unforeseen situations—while still offering competitive performance and greater scalability than centralized approaches.

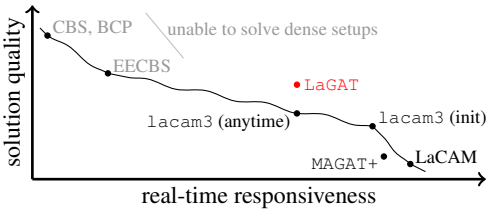


Figure 1: Pareto frontier consisting of state-of-the-art algorithms in dense MAPF instances.

While both fields have made notable progress, learning-based approaches still lag significantly behind their search-based counterparts. Across *all* aspects, including scalability, leading search-based planners such as `lacam3` (Okumura 2024) consistently and significantly outperform existing learning paradigms (Skrynnik et al. 2025; Andreychuk et al. 2025). At the same time, as seen in various domains—most notably the game of Go (Silver et al. 2016, 2017)—the combination of search and learning has proven to be a powerful strategy, suggesting that a similar integration could lead to more capable MAPF methods. Indeed, numerous efforts have been made to enhance classical MAPF search algorithms using neural components (Yu et al. 2023; Veerapaneni et al. 2024; Yan and Wu 2024; Wang et al. 2025; Alam et al. 2025). These approaches have succeeded in improving specific aspects of base planners; however, when considering factors such as real-time responsiveness, performance stability, scalability, and comparison with state-of-the-art planners, their impact remains limited—providing insight but not competitive performance. One of these even concluded that *"all learnt MAPF works require non-trivial data collection and complex models but perform worse than LaCAM"* (Veerapaneni et al. 2024). Another benchmark paper similarly reported that *"our results demonstrate that current learning-based methods fail to exhibit a clear advantage over simple rule-based heuristics"* (Tan et al. 2025).

These insights raise practical questions in MAPF: *Can neural components actually provide benefits over state-of-the-art search-based planners, and if so, how?* In this study, we demonstrate that, in densely populated MAPF instances,

*These authors contributed equally.

a hybrid approach combining search and learning can outperform purely search-based methods in terms of solution quality. Our approach, named LaGAT, involves a computational overhead due to neural network inference; however, we find that a substantial improvement in solution quality justifies this overhead. As illustrated in Fig. 1, this work marks a new frontier—*arguably for the first time by neural methods*—surpassing the existing Pareto frontier established by leading search-based algorithms. Together with the recent evolution of the hybrid planner in lifelong MAPF (Jiang et al. 2025)—where tightly coupled coordination is not critical unlike dense one-shot MAPF setups—our findings provide a strong incentive to further pursue this direction.

LaGAT embeds MAGAT+, a lightweight neural heuristic, into the search-based MAPF algorithm *LaCAM* (Okumura 2023b). MAGAT+ is our improved version of MAGAT (Li et al. 2021b), a seminal decentralized neural MAPF policy, attaining similar performance to leading neural methods while running faster and requiring less data to train. We first pre-train MAGAT+ by imitating lacam3 trajectories, then fine-tune it on the map of interest using a small amount of data. Although the one-off adaptation might appear to constrain flexibility, it is a logical strategy in practice, given that major MAPF applications (e.g., warehouse automation and railway scheduling) assume static map layouts. As neural policies alone are prone to inaccuracies and deadlocking, we cover for MAGAT+'s weaknesses using *(i) PIBT* collision shielding (Okumura et al. 2022; Veerapaneni et al. 2024) and *(ii)* a novel deadlock detection mechanism. The resulting LaGAT surpasses the real-time performance of lacam3 in dense setups.

2 Preliminaries

Our proposed LaGAT is built upon PIBT and LaCAM—two high-performing algorithms that currently lead research on real-time and scalable MAPF—together with MAGAT. Starting from the problem formulation, this section outlines PIBT, LaCAM, and neural MAPF policies, including MAGAT, which serve as the foundation for the rest of this paper.

2.1 Problem Definition and Notations

We consider a widely used MAPF formulation (Stern et al. 2019), defined by a team of agents $A = \{1, 2, \dots, n\}$ 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: two agents cannot occupy the same vertex simultaneously, and two agents cannot swap their occupied vertices within one timestep. Then, for each agent $i \in A$, we aim to assign a collision-free path, denoted by $\pi_i = (v^0 = s_i, v^1, \dots, v^T = g_i)$. The solution quality is assessed by *sum-of-costs* (SoC; aka. flowtime), which sums the travel time of each agent until it stops at the target location and no longer moves.

Herein, a *configuration* $\mathcal{Q} \in V^n$ refers to the locations for all agents. We use $\perp$ as an "undefined" sign. The dist function returns the shortest path length between two vertices, while $\mathsf{neigh}(v)$ represents adjacent vertices for $v \in V$.

2.2 PIBT and LaCAM

PIBT (priority inheritance with backtracking) (Okumura et al. 2022) is a computationally lightweight function to generate a *transitionable* configuration $\mathcal{Q}'$ given another configuration $\mathcal{Q}$, where each agent i can move from the current location $\mathcal{Q}[i]$ to the next $\mathcal{Q}'[i]$ within one action, without colliding. It has gained notable popularity as fast configuration generation is central to MAPF (Jiang et al. 2025; Yukhnevich and Andreychuk 2025). For each generation, PIBT needs a *preference* for each agent i, a sorted list of candidate actions $v \in \mathsf{neigh}(\mathcal{Q}[i]) \cup \{\mathcal{Q}[i]\}$. This is arranged in an ascending order with $\mathsf{dist}(v, g_i)$, called *cost-to-go*, which encourages agents to head towards their respective goals.

PIBT alone functions only as a greedy collision avoidance planner, often leading to deadlocks or livelocks. *LaCAM (lazy constraints addition search)* (Okumura 2023b) improves upon this by adding a systematic search over configurations, which derives a transitionable configuration sequence that connects the start and goal. As enumerating all successor configurations requires exponential time for the number of agents, LaCAM uses PIBT to guide the search, creating only one successor for each search node expansion. Other successors are gradually synthesized by adding *constraints*, which prescribe the next location of each agent. This lazy successor generation significantly reduces planning efforts, rendering LaCAM a reliable, real-time, and scalable MAPF solver (Zhang et al. 2024a; Shankar, Okumura, and Prorok 2025).

Algorithm 1 presents a simplified pseudocode of LaCAM; disregard Line 15 for now, which will be revisited in Sec. 3.3. As with most search methods, it proceeds the search by updating two data structures: *(i)* an $Open$ stack that stores search nodes, denoted as $\mathcal{N}$, and *(ii)* an $Explored$ table that stores already discovered configurations. Each search node also maintains constraints, denoted as $\mathcal{C}$, that are used for constrained configuration generation with PIBT (Line 10). Initially, the constraints include nothing (denoted $\mathcal{C}^{\text{init}}$), allowing PIBT to generate an arbitrary configuration. They are gradually developed for each node invocation (Line 9; abstracted for brevity), eventually specifying exactly one configuration without freedom. Once the search hits the goal, LaCAM extracts a solution by backtracking ancestor nodes (Line 6); otherwise, $Open$ eventually gets empty, signifying the unsolvability of the instance (Line 16).

LaCAM is a *complete* algorithm for MAPF, i.e., it finds a solution if the problem is solvable; otherwise, it reports the non-existence. Subsequent work (Okumura 2023a) shows that slight adaptations can make it an anytime version *LaCAM** that eventually converges on an optimal solution.

2.3 Preference Construction

Although LaCAM has advanced the frontier of scalable MAPF, its solution is known to be highly suboptimal (Shen et al. 2023). This primarily stems from the myopic nature of PIBT, which lacks the ability to reason about long-term coordination beyond one timestep. While LaCAM's constraints help prevent local miscoordination such as deadlocks, they do not promote (near-)optimal coordination.

Algorithm 1: LaCAM for MAPF

input: graph G, agents A, starts $(s_i)^{i \in A}$, goals $(g_i)^{i \in A}$
output: solution or `NO_SOLUTION`

initialize $Open$, $Explored$
$\mathcal{N}^{\text{init}} \leftarrow \langle config : (s_1, \dots, s_n),\ constraints : [\![\ \mathcal{C}^{\text{init}}\ ]\!] \rangle$
$Open$.push($\mathcal{N}^{\text{init}}$); $Explored[\mathcal{S}] = \mathcal{N}^{\text{init}}$
while $Open \neq \emptyset$ **do**
  $\mathcal{N} \leftarrow Open$.top()
  if $\mathcal{N}.config = (g_1, \dots, g_n)$ **then return** backtrack($\mathcal{N}$)
  if $\mathcal{N}.constraints = \emptyset$ **then** $Open$.pop(); **continue**
  $\mathcal{C} \leftarrow \mathcal{N}.constraints$.pop()
  update_constraints($\mathcal{N}, \mathcal{C}$) ▷ *constraints synthesis*
  $\mathcal{Q}^{\text{new}} \leftarrow$ configuration_generator($\mathcal{N}, \mathcal{C}$) ▷ *call PIBT*
  if $\mathcal{Q}^{\text{new}} = \perp$ **then continue**
  if $Explored[\mathcal{Q}^{\text{new}}] \neq \perp$ **then continue**
  $\mathcal{N}^{\text{new}} \leftarrow \langle config : \mathcal{Q}^{\text{new}},\ constraints : [\![\ \mathcal{C}^{\text{init}}\ ]\!] \rangle$
  $Open$.push($\mathcal{N}^{\text{new}}$); $Explored[\mathcal{Q}^{\text{new}}] = \mathcal{N}^{\text{new}}$
  DEADLOCKDETECTION($\mathcal{N}$, $\mathcal{Q}^{\text{new}}$) ▷ *for neural policies*
return `NO_SOLUTION`

For more capable MAPF planners, it is crucial to embed advanced coordination reasoning in PIBT, corresponding to devising the preference construction currently determined by cost-to-go. This leads to a line of research, such as congestion mitigation (Chen et al. 2024; Zhang et al. 2024b) and tiebreaking (Okumura and Nagai 2025). Among these, a promising strategy is to leverage offline experience to construct the preferences, i.e., imitation learning from (near-)optimal MAPF solutions that were collected in a computationally intensive manner (Veerapaneni et al. 2024, 2025; Jiang et al. 2025). This technique is also called *collision shielding (CS-PIBT)* because it uses PIBT to protect the neural policy execution from inter-agent collisions.

This study also falls in the last category. Within this scheme, prior work (Veerapaneni et al. 2024) concluded that neural-enhanced LaCAM has difficulty outperforming purely heuristic counterparts due to imperfections in learned models and slower inference speeds. In this work, however, we overturn this claim through careful architectural design and implementation of the neural model, search, and target environments.

2.4 Neural Decentralized MAPF Policies

Alongside the development of search-based MAPF methods, the recent machine learning revolution fueled by neural networks has also advanced learning-based approaches to MAPF (Alkazzi and Okumura 2024). In particular, motivated by scalable and adaptive deployments, research on decentralized MAPF policies has gained notable momentum. Typically, these approaches aim to design a neural policy that interprets surrounding information—represented as a field of view (FOV)—and that selects the next action within a grid-world context, optionally assuming inter-agent communication. Such policies are commonly trained using reinforcement learning (Sartoretti et al. 2019), imitation learning (Andreychuk et al. 2025; Jiang et al. 2025), or a combination of both. Graph neural networks (GNNs) are sometimes adopted as the backbone, as they naturally capture agent interactions through their message-passing structure. Our study builds on a representative GNN architecture for MAPF, known as *MAGAT (message-aware graph attention network)* (Li et al. 2021b).

2.5 MAGAT

A GNN-based MAPF planner receives a local observation for each agent i, encoded as a node feature $\mathbf{x}_i$ (optionally processed by a CNN), and operates over a communication graph $\mathcal{G}$, constructed based on spatial proximity within a radius $R^{\text{comm}} \in \mathbb{N}_{>0}$. Each agent i transmits a latent message $\mathbf{m}_i$ over $\mathcal{G}$, receives messages $\mathbf{m}_j$ from neighboring agents j, and computes its action based on the aggregated information. This process is represented by a single GNN layer. Finally, a multi-layer perceptron (MLP) decodes the embedding into an action distribution.

While earlier GNN-based policies, e.g., (Li et al. 2020), aggregate incoming messages uniformly, MAGAT introduces a message-dependent attention mechanism that enables each agent i to weigh the importance of messages from its neighbors j. This prioritization improves generalization to larger grids and higher agent densities.

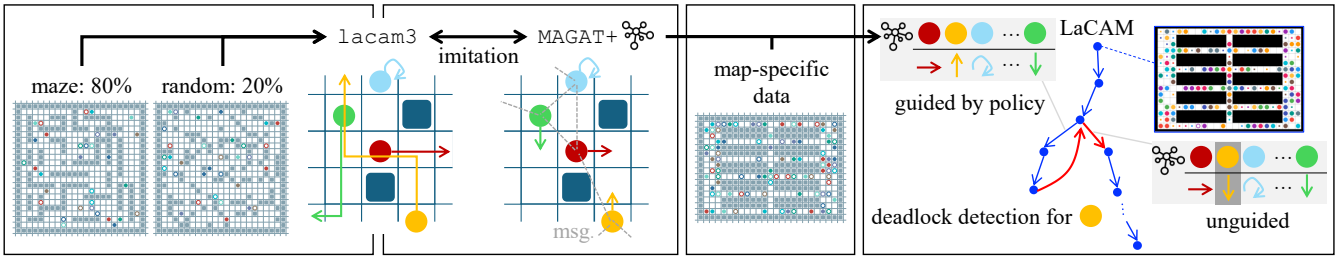
3 LaGAT

The proposed `LaGAT` is a *map-specific*, complete, search-based MAPF solver that leverages neural policies as heuristics. It integrates an enhanced version of LaCAM with an improved MAGAT policy, termed `MAGAT+`, which injects long-horizon coordination into PIBT preferences through learned interaction reasoning. We use MAGAT as our backbone because of its ability to capture complex agent interactions, which is paramount in dense setups. As illustrated in Fig. 2, `LaGAT` consists of four stages: *(i)* collecting expert demonstration data using `lacam3`, *(ii)* pretraining `MAGAT+`, *(iii)* fine-tuning `MAGAT+` to the target environment, and *(iv)* executing LaCAM search guided by `MAGAT+`, augmented with a newly developed deadlock detection mechanism. In what follows, we describe the neural policy preparation and its integration into LaCAM in detail.

3.1 Neural Policy Architecture (`MAGAT+`)

Local Observation. For each configuration $\mathcal{Q}$, each agent i receives an observation tensor o_i centered 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 first three channels are binary matrices indicating: *(i)* the presence of obstacles, *(ii)* the presence of other agents, and *(iii)* a projection of the goal direction. This design follows the observation architecture commonly used in neural MAPF policies (Sartoretti et al. 2019; Alkazzi and Okumura 2024). The fourth channel, an addition to MAGAT, encodes cost-to-go values: for each location $v \in V$ within the FOV, the value is computed as $(\mathsf{dist}(v, g_i) - \mathsf{dist}(\mathcal{Q}[i], g_i))/(2R^{\text{obs}})$.

Edge Feature. In the communication graph $\mathcal{G}$, each edge (i, j) is associated with a three-dimensional feature vector ω_{ij}, consisting of the relative position (in x and y) and the Manhattan distance between the agents. While the original



(a) expert data collection (b) pre-training (c) fine-tuning (d) search

Figure 2: Overview of LaGAT. We utilize 21K instances for pre-training (a, b) and 1K instances for fine-tuning (c). The resulting policy MAGAT+ is embedded into LaCAM search over configurations (d). When a deadlock is detected for a specific agent, the search performs a rollback, and the neural guidance is selectively overridden by the default, non-learning PIBT mechanism.

MAGAT does not incorporate edge features, we empirically found that including them improves the model capability.

Model Architecture. MAGAT+ is an imitation learning 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 $\mathcal{G}$. The model follows a similar design to the original MAGAT, consisting of *(i)* a CNN encoder that converts o_i into the node feature $\mathbf{x}_i$, *(ii)* GNN layers, and *(iii)* an MLP decoder. Unlike the original MAGAT, which uses a single graph attention layer, MAGAT+ employs three stacked layers, to enable wider communication and greater cooperation via more message-passing interactions. Another key modification is that these layers are extended to incorporate edge features, enabling richer interaction modeling. Concretely, let $\mathbf{x}_i^{(l)}$ denote the node feature of agent i at layer l, and ω_{ji} the edge feature from agent j to i. The layer update proceeds with:

$$\mathbf{x}_i^{(l+1)} = \sigma\left(\mathbf{W}_{\mathrm{R}}^{(l)}\,\mathbf{x}_i^{(l)} + \mathbf{m}_i^{(l)}\right) \tag{1}$$

$$\mathbf{m}_i^{(l)} = \sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(l)}\left(\mathbf{W}_n^{(l)}\,\mathbf{x}_j^{(l)} + \mathbf{W}_e^{(l)}\,\mathbf{w}_{ji}\right) \tag{2}$$

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\mathrm{LeakyReLU}\left(a_{ij}^{(l)}\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(a_{ik}^{(l)}\right)\right)} \tag{3}$$

$$a_{ij}^{(l)} = \left(\mathbf{x}_i^{(l)}\right)^{\top}\left(\Theta_n^{(l)}\,\mathbf{x}_j^{(l)} + \Theta_e^{(l)}\,\mathbf{w}_{ji}\right) \tag{4}$$

Here, $\mathbf{w}_{ji} = \phi(\omega_{ji})$ is the processed edge feature via an MLP ϕ, α_{ij} is the normalized attention weights, $\mathbf{W}_{\{\mathrm{R,n,e}\}}^{(l)}$ and $\Theta_{\{n,e\}}^{(l)}$ are the learnable weights, and σ is a non-linearity. These enhancements allow GNN to capture more nuanced relational structures, improving policy performance in dense, complicated MAPF scenarios.

3.2 Training Pipeline

MAPF Instance Preparation. We use the POGEMA toolkit (Skrynnik et al. 2025), also employed in the development of MAPF-GPT (Andreychuk et al. 2025), a state-of-the-art imitation learning policy for MAPF. Following MAPF-GPT's training protocol, we generate a total of 21K instances. 20% feature randomly placed obstacles, while the remaining 80% are maze-like environments (see Fig. 2A). Map sizes range from 17×17 to 21×21, and each instance includes 16, 24, or 32 agents. For reference, MAPF-GPT uses 3.75M instances for training.

Collecting Expert Trajectories. To generate demonstration trajectories for training, we employ lacam3 (Okumura 2024) instead of the conflict-based search variant (Barer et al. 2014) used in MAGAT. lacam3 is an anytime planner that enables large-scale trajectory collection with low computational cost. We adopt a staged timeout strategy with time limits of $[1, 5, 15, 60]$s. If no solution is found within a given limit, the planner is re-run with the next longer timeout. As the training set does not include challenging situations, most instances were solved within 1 s.

Pre-Training. MAGAT+, with $R^{\mathrm{comm}} = 7$ and $R^{\mathrm{obs}} = 5$, is pre-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 optimizer (Loshchilov and Hutter 2019). Additional training details are provided in the appendix. To mitigate the distributional shift common in imitation learning, we further apply on-demand dataset aggregation (Ross, Gordon, and Bagnell 2011), following Li et al. (2020).

Map-wise Fine-Tuning. Given a target map G (e.g., a warehouse map in Fig. 2), we generate additional 1,000 instances on G and corresponding lacam3's trajectories, this time with higher agent densities: 32, 48, 64, and 80 agents. These instances are used to fine-tune the pre-trained model, specializing it for G. The fine-tuning process runs for 52 epochs and typically completes within 4–8 hours.

3.3 Search with Imperfect Neural Policies

Neural MAPF policies alone, including MAGAT+, are insufficient for solving MAPF, as their outputs are not guaranteed to be collision-free and therefore require additional safeguard mechanisms. Moreover, such policies lack theoretical guarantees—such as completeness—which are often critical in real-world MAPF applications. Integrating a non-learning search component is thus both practical and effective in addressing these limitations.

Algorithm 2: DEADLOCKDETECTION

input: search node $\mathcal{N}$, generated configuration $\mathcal{Q}^{\text{new}}$
params: depth of deadlock detection $d \in \mathbb{N}_{\geq 0}$

```
1: N^ans ← parent(N)                                  ▷ Q^ans := N^ans.config
2: for 1, 2, ..., d do until N^ans = ⊥
3:     for i ∈ A s.t. Q^new[i] ≠ g_i ∧ Q^new[i] = Q^ans[i] do
4:         if adj(Q^new, i) = adj(Q^ans, i) then
5:             N^ans.unguided.insert(i)
6:     if N^ans.unguided has updated then
7:         N^ans.constraints ← ⟦ C^init ⟧              ▷ reset constraints
8:         Open.push(N^ans)                            ▷ node reinsert
9:     N^ans ← parent(N^ans)
```

To this end, we adopt the strategy proposed by Veerapaneni et al. (2024), which uses PIBT to enforce safety and performs LaCAM on top. In this setup, agent preferences in PIBT are constructed by sorting candidate vertices in descending order of the probabilities predicted by MAGAT+. Unlike the naïve execution of neural policies that sample actions *probabilistically*, this framework uses the model prediction *deterministically* during the search.

Although our work builds on this concept, empirical results suggest that the vanilla neuro-enhanced approach does not fully realize the model's potential to surpass existing MAPF solvers. Several factors contribute to this limitation: *(i)* Large models, such as those used in language modeling or MAPF-GPT,[1] are unsuitable for quick search due to their slow inference speed; *(ii)* As a result, compact neural networks must be used. However, these lack sufficient representational power to mimic expert behavior accurately; *(iii)* Inaccurate predictions lead to deadlocks or livelocks, especially with deterministic sampling employed, which significantly degrades LaCAM performance; *(iv)* While probabilistic sampling improves robustness, it tends to execute suboptimal actions, reducing overall solution quality and limiting the benefits of hybridization.

In summary, *we need to use neural policies deterministically, but they can misbehave occasionally*. This motivates a systematic override mechanism that replaces model outputs with the default PIBT behavior when misbehavior is detected, which we refer to as *deadlock detection*.

3.4 Deadlock Detection

A common failure of neural policies under deterministic sampling arises when an agent repeatedly executes the same action sequence without making progress toward its goal. For example, we often observe oscillatory behavior between two positions, resulting in a livelock-like state.

LaCAM allows us to systematically detect such failure cases by backtracing recent agent-wise state histories. Given a hyperparameter $d \in \mathbb{N}_{\geq 0}$ (typically less than 3), we check whether an agent has revisited the same state within the past d timesteps. If repetition is detected, the agent is deemed to be in deadlock or livelock, and its neural output is overridden with the default, non-learned preference.

[1]The smallest (2M) and largest (85M) MAPF-GPT models require 50 and 280 ms per inference for 128 agents (including feature construction, CPU-GPU data transfer), respectively. In contrast, MAGAT+ has 760K parameters and runs in only $\leq$5 ms for the same setup.

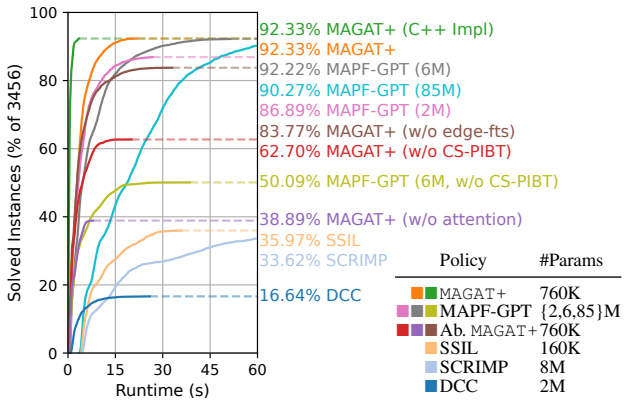


Figure 3: Evaluation on neural policies by success rate and runtime. Model sizes are listed in the bottom right corner.

Algorithm 2 crystallizes this idea, embedded at Line 15 in Alg. 1. For each search node in LaCAM, we maintain a set of *unguided* agents by neural policies, initialized with $\emptyset$. When an agent i is detected to be in deadlock at a given configuration, it is added to the unguided set, thereby forcing it to follow the default preference. The detection is performed at Lines 3 and 4, by comparing its next location $\mathcal{Q}^{\text{new}}[i] \in V$ with the corresponding location in the ancestor state $\mathcal{Q}^{\text{ans}}[i]$, based on the following three conditions: *(i)* the agent has not yet reached its goal, *(ii)* the location remains unchanged, and *(iii)* the vicinity remains the same, defined as: $\mathsf{adj}(\mathcal{Q}, i) = \{(v, j) \mid v \in \mathsf{neigh}(\mathcal{Q}[i]), \mathcal{Q}[j] = v\}$. Once the unguided set is updated, Alg. 2 reinserts the corresponding node into *Open*, allowing the next search iteration to explore it (Line 8). In addition, we discard the constraints explored so far (Line 7) to eliminate unnecessary suboptimal behavior forced by the constraints.

Empirically, we will see that Alg. 2 is essential for achieving high-performance hybrid MAPF planning. Theoretically, it preserves the key properties of LaCAM:

Theorem 1. *LaCAM with Alg. 2 is complete.*

Proof sketch. According to (Okumura 2023b), LaCAM is complete regardless of the preferences in PIBT. Algorithm 2 does not violate the structural assumptions of LaCAM, except for Line 7, which resets constraints. For each search node, once an agent $i \in A$ is added to the *unguided* set, it remains *unguided* for the rest of the search. Since A is finite, the *unguided* set cannot grow infinitely, and consequently, constraints cannot be reset infinitely often. This ensures that, from any given configuration, all successor configurations will eventually be explored during the search, thereby establishing completeness. □

The eventually optimal variant, LaCAM*, remains readily attainable too. The corresponding proof follows directly from (Okumura 2023a).

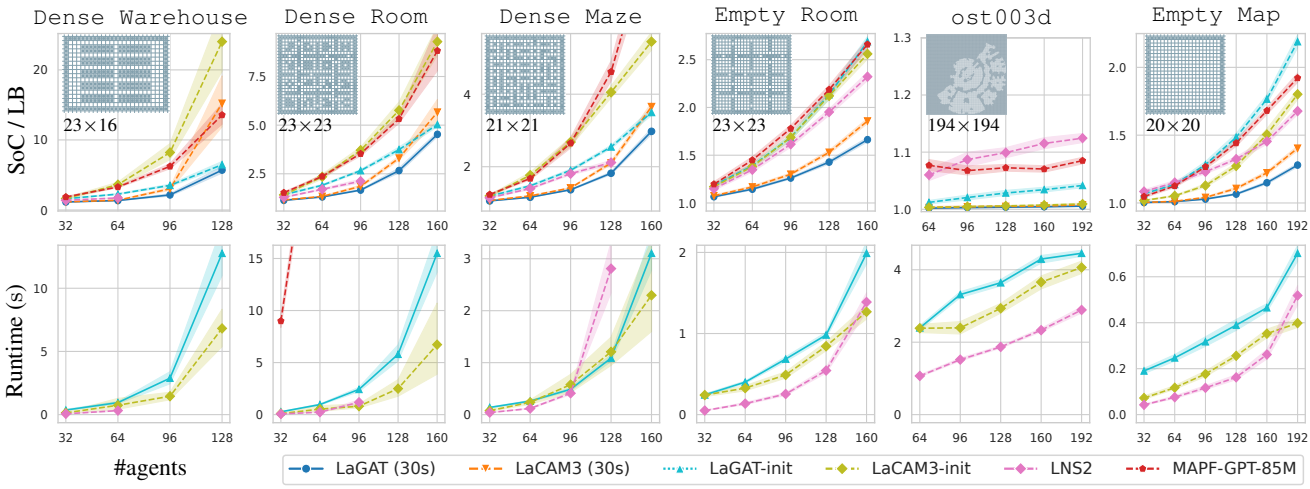


Figure 4: Map-wise evaluation. Runtime and sum-of-costs scores are average within solved instances for each method. Transparent regions represent 95% confidence intervals. SoC is normalized by its trivial lower bound $\sum_{i \in A} \mathsf{dist}(s_i, g_i)$. `MAPF-GPT` is significantly slower than the other methods and falls outside the plotted runtime range. For reference, we also display the results of another leading search-based solver, LNS2 (Li et al. 2022). With a timeout of 30 s, LNS2 fails in dense setups (e.g., 0% in `Dense Warehouse` with 128 agents), so we only plot LNS2 data points with a success rate higher than 80%.

4 Evaluation

We contrast `LaGAT` with various state-of-the-art planners, comprising search-based solvers and neural policies based on either imitation or reinforcement learning:

- `lacam3` (Okumura 2024), one of the most efficient, scalable and reliable search-based MAPF solvers available today.
- `MAPF-GPT` (Andreychuk et al. 2025), the most capable imitation-learning policy to date, using GPT-like architecture. The authors provided three model sizes: 2M, 6M, and 85M. Generally, larger models are more capable.
- `SSIL` (Veerapaneni et al. 2025), a follow-up of (Veerapaneni et al. 2024) that uses GNN-based imitation learning.
- `SCRIMP` (Wang et al. 2023) and `DCC` (Ma, Luo, and Pan 2021), reinforcement learning-based representatives.
- `MAGAT+`: our enhanced MAGAT model, without map-specific fine-tuning or search integration.

All implementations are obtained from the authors' public repositories or the POGEMA benchmark suite (Skrynnik et al. 2025). `MAGAT+` is developed and trained in Python, and translated into C++ for faster inference. `LaGAT` is also coded in C++. Unless otherwise noted, all imitation learning methods (`MAPF-GPT`, `SSIL`, and `MAGAT+`) are equipped with advanced collision shielding (CS-PIBT), as recommended in (Veerapaneni et al. 2025).

4.1 Neural Policy Performance

We begin by evaluating the standalone performance of neural policies to justify the use of `MAGAT+` as the learned component in `LaGAT`. Figure 3 shows a cactus plot illustrating the number of instances solved within varying time budgets, up to 60 s, across different map types and agent densities. Specifically, we evaluate 3,456 instances generated using POGEMA (Skrynnik et al. 2025), covering six map types: `Warehouse`, `Room`, `Maze`, and their denser variants. Further details about these maps are available in the appendix. For each map, we consider four agent densities, ranging from 32 to 160 agents. All experiments are conducted on a computing cluster equipped with Intel Xeon Platinum 8452Y CPUs and NVIDIA L40S GPUs for fast neural inference. We also include several ablated versions of `MAGAT+` to see their functionality.

From Fig. 3, we first observe that imitation learning-based policies consistently outperform those trained via reinforcement learning. The incorporation of CS-PIBT further boosts their effectiveness, aligning with prior findings (Andreychuk et al. 2025; Veerapaneni et al. 2025). Notably, `MAGAT+` achieves success rates comparable to `MAPF-GPT`, despite having a significantly smaller model size and being trained on a smaller dataset. This strong performance is attributed to its carefully designed graph attention architecture. Ablated versions confirm that each architectural component contributes meaningfully to the model's overall capability. Furthermore, `MAGAT+` excels in inference speed; particularly in our optimized C++ version, which significantly outperforms other models. This efficiency is a critical feature for integrating neural policies into search-based MAPF solvers. Armed with these insights, we next challenge `lacam3`.

4.2 `LaGAT` vs. `lacam3`

As `lacam3` is an anytime algorithm that progressively improves solution quality over time, we evaluate the planners' real-time performance based on two criteria: *(i)* quality of the initial feasible solution, and *(ii)* final solution quality at the deadline, set to 30 s. Since `LaGAT` alone does not support anytime refinement, we incorporate a widely used post-

processing method, large neighborhood search (LNS) (Li et al. 2021a; Okumura, Tamura, and Défago 2021), which iteratively selects a subset of agents and refines their paths. We adopt the same LNS implementation used in `lacam3` and attach it to `LaGAT` to ensure a fair comparison.

Our evaluation includes both constrained challenging environments (e.g., `Dense Warehouse/Room`) and sparse, relatively easy ones (e.g., `ost003d`). For each map and each density, 32 instances are prepared. The experiment is conducted on a cluster with Intel Xeon Gold 6248R processors and NVIDIA GeForce RTX 2080 Ti GPUs.

Figure 4 compares `LaGAT` with leading search-based and learning-based solvers, namely `lacam3` and `MAPF-GPT`. All plotted results have a nearly 100% success rate. In dense environments (first three columns), although `LaGAT` incurs non-negligible runtime for deriving initial solutions, it consistently produces significantly higher-quality initial solutions than not only `MAPF-GPT`, but also `lacam3`. As a result, it achieves markedly better final performance by the time limit—*completely outperforming `lacam3`*. Figure 5 demonstrates this conclusion. The appendix contains success rates, and empirical results for additional maps.

We note that this advantage does not extend universally. On larger, sparser maps such as `ost003d`, `lacam3` reliably generates near-optimal solutions, while `LaGAT` tends to yield slightly suboptimal initial solutions. Nonetheless, thanks to the efficiency of LNS in sparse settings, `LaGAT` ultimately achieves comparable final performance. Interestingly, we observe that `LaGAT` can even slightly outperform in most cases; we presume that `LaGAT` solutions exhibit structural regularities that LNS can exploit more effectively.

4.3 Analyzing Roles of Learning and Search

To better understand which technical components are responsible for the strong performance of initial solutions in the dense setups, the following ablated versions of `LaGAT` are tested on the same experimental setup as in Sec. 4.2: *(i) w/o map-specific fine-tuning*: uses the pretrained policy without additional adaptation to the target map; *(ii) w/o pre-training*: uses `MAGAT+` trained only on data from the target map, without general pre-training; *(iii) w/o neural guidance*: reverts to the original LaCAM; *(iv) w/o deadlock detection*: disables Alg. 2; *(v) w/o search*: i.e., `MAGAT+` with CS-PIBT.

Figure 6, evaluated on the densest scenarios for the first three maps in Fig. 4, demonstrates that all components are essential for simultaneously achieving high success rates and low solution costs, underscoring the effectiveness of `LaGAT`'s design. In particular, both pre-training and fine-tuning are critical for improving solvability, while neural guidance and search mechanisms, especially deadlock detection, contribute significantly to reducing solution cost.

5 Discussion

This study demonstrates that combining search with neural policies is effective in dense, highly congested MAPF scenarios, where purely search-based methods alone struggle to achieve strong results.

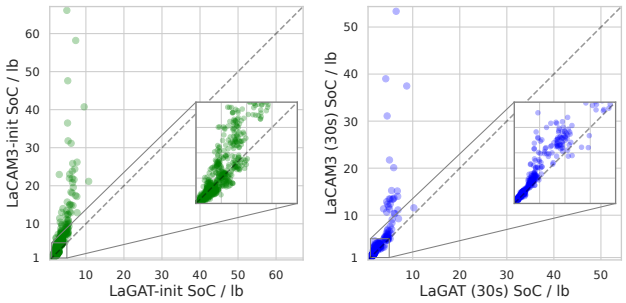


Figure 5: Instance-wise sum-of-costs comparison between `lacam3` and `LaGAT` across all instances included in Fig. 4. Points in the upper-left triangular region indicate where `LaGAT` outperforms `lacam3`.

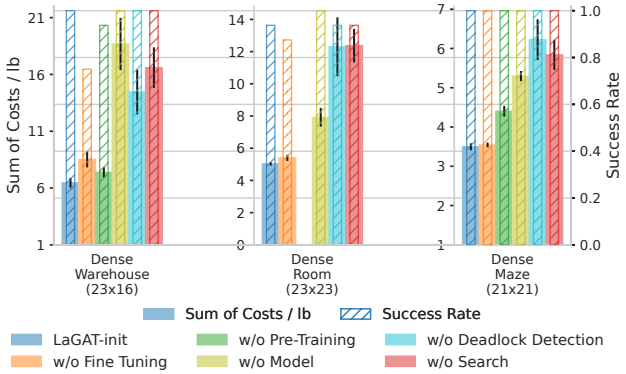


Figure 6: Ablation study on the densest scenarios in Fig. 4.

Why does LaGAT succeed? While prior work has attempted to embed neural policies into configuration-based planners—most notably Veerapaneni et al. (2024), who integrated MAGAT into LaCAM—the reported performance gains did not justify the runtime cost. In contrast, our results demonstrate that, neural guidance can yield clear advantages in specific scenarios, with the following key insights:

- *Pre-train a general policy, then post-train task-specific variants:* This widely adopted strategy has proven effective in domains such as language modeling (Devlin et al. 2019) and robotics (Black et al. 2024). Fine-tuning enables policies to specialize for deployment environments, achieving performance not attainable with general models alone. We demonstrate that the same principle is crucial in MAPF, where the structural characteristics of each map strongly influence policy effectiveness.
- *Override policy outputs if necessary:* Neural MAPF policies are inherently imperfect and may lead to deadlocks or livelocks in practice. Fortunately, such pathological behaviors can be readily detected by examining recent state histories. A fallback is always available when detected, allowing for a return to standard MAPF methods without neural guidance. This corrective layer, largely overlooked in prior studies, proves essential for achieving high performance in hybrid approaches.

- *Imitation that scales:* In our dense scenarios, while `lacam3` quickly produces near-optimal trajectories for lower agent densities, it cannot reach the same quality for higher agent density instances within the 30 s budget. `MAGAT+` is trained on the near-optimal trajectories generated for the lower agent densities and is able to learn the rules which generalize to higher agent densities, even where the teacher itself falters. We hypothesize that, using the guidance from `MAGAT+`, `LaGAT` is able to visit higher-quality states, leading to better solutions than `lacam3` in the densest challenging setups.

Combined with other carefully engineered aspects, `LaGAT` redefines the Pareto frontier that marks the speed–quality tradeoff established by leading search-based solvers.

When is LaGAT appealing? Current `LaGAT` excels in scenarios that meet the following conditions: *(i)* relatively small maps (e.g., 20×20), *(ii)* obstacle-rich environments, and *(iii)* high agent density. For larger or less constrained maps, mere greedy collision-avoidance planners (like PIBT) along with global awareness (Zhang et al. 2024b; Kato et al. 2025) are often sufficient to achieve near-optimal performance. For maps much larger than the communication and observation ranges of decentralized policies, like `MAGAT+`, the local nature of these policies can lead to suboptimal solutions, leaving room for future investigation in this area. Moreover, agent density is a critical factor in determining instance difficulty: when the density is low, advanced MAPF solvers can already find solutions efficiently, leaving little room for neural components to offer practical benefits given their additional inference overhead.

Solidifying Neuro-Guided Multi-Agent Search. Our study gives positive evidence that combining search with neural guidance can achieve frontline performance in MAPF. `LaGAT` has two more desirable properties: *(i)* unlike purely learned approaches, it offers completeness guarantees, and *(ii)* the neural heuristic can be swapped in plug-and-play fashion, depending on the application. For these reasons, we believe this neuro-guided approach, where decentralized neural policies guide search, is a promising and flexible foundation for addressing broader classes of multi-agent coordination problems beyond MAPF.

Acknowledgments

This research was funded in part by Trinity College Cambridge, European Research Council (ERC) Project 949940 (gAIa), JST ACT-X (JPMJAX22A1), and JST PRESTO (JPMJPR2513).

References

Alam, M. A.; Mahmud, S.; Mamun-Or-Rashid, M.; and Khan, M. M. 2025. Optimizing Node Selection in Search Based Multi-Agent Path Finding. *Autonomous Agents and Multi-Agent Systems (JAAMAS)*.

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. 2025. MAPF-GPT: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint*.

Chen, Z.; Harabor, D.; Li, J.; and Stuckey, P. J. 2024. Traffic flow optimisation for lifelong multi-agent path finding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Jiang, H.; Wang, Y.; Veerapaneni, R.; Duhan, T.; Sartoretti, G.; and Li, J. 2025. Deploying Ten Thousand Robots: Scalable Imitation Learning for Lifelong Multi-Agent Path Finding. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Kato, T.; Okumura, K.; Sasaki, Y.; and Yokomachi, N. 2025. Congestion Mitigation Path Planning for Large-Scale Multi-Agent Navigation in Dense Environments. *IEEE Robotics and Automation Letters (RA-L)*.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P.; and Koenig, S. 2021a. Anytime multi-agent path finding via large neighborhood search. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

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 AAAI Conference on Artificial Intelligence (AAAI)*.

Li, Q.; Gama, F.; Ribeiro, A.; and Prorok, A. 2020. Graph Neural Networks for Decentralized Multi-Robot Path Planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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 (RA-L)*.

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

Ma, Z.; Luo, Y.; and Pan, J. 2021. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters (RA-L)*.

Okumura, K. 2023a. Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Okumura, K. 2023b. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Okumura, K. 2024. 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)*.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding. *Artificial Intelligence (AIJ)*.

Okumura, K.; and Nagai, H. 2025. Lightweight and Effective Preference Construction in PIBT for Large-Scale Multi-Agent Pathfinding. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Okumura, K.; Tamura, Y.; and Défago, X. 2021. Iterative Refinement for Real-Time Multi-Robot Path Planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Prorok, A.; Blumenkamp, J.; Li, Q.; Kortvelesy, R.; Liu, Z.; and Stump, E. 2021. The holy grail of multi-robot planning: Learning to generate online-scalable solutions from offline-optimal experts. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. 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)*.

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 (RA-L)*.

Shankar, A.; Okumura, K.; and Prorok, A. 2025. LF: Online Multi-Robot Path Planning Meets Optimal Trajectory Control. *arXiv preprint arXiv:2507.11464*.

Shen, B.; Chen, Z.; Cheema, M. A.; Harabor, D. D.; and Stuckey, P. J. 2023. Tracking progress in multi-agent path finding. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

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*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature*.

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

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Tan, J.; Luo, Y.; Li, J.; and Ma, H. 2025. Reevaluation of Large Neighborhood Search for MAPF: Findings and Opportunities. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Veerapaneni, R.; Jakobsson, A.; Ren, K.; Kim, S.; Li, J.; and Likhachev, M. 2025. 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)*.

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 International Conference on Automated Planning and Scheduling (ICAPS)*.

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 AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, Y.; Xiang, B.; Huang, S.; and Sartoretti, G. 2023. Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Yan, Z.; and Wu, C. 2024. Neural neighborhood search for multi-agent path finding. In *Proceedings of the International Conference on Learning and Representation (ICLR)*.

Yu, C.; Li, Q.; Gao, S.; and Prorok, A. 2023. Accelerating multi-agent planning using graph transformers with bounded suboptimality. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Yukhnevich, E.; and Andreychuk, A. 2025. Enhancing PIBT via multi-action operations. In *League of Robot Runners Expo*.

Zhang, Y.; Chen, Z.; Harabor, D.; Le Bodic, P.; and Stuckey, P. J. 2024a. Planning and execution in multi-agent path finding: Models and algorithms. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2024b. Guidance graph optimization for lifelong multi-agent path finding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.