RAILGUN: A Unified Convolutional Policy for Multi-Agent Path Finding Across Different Environments and Tasks (Extended Abstract)

Yimin Tang[1*], **Xiao Xiong**[2*], **Jingyi Xi**[2], **Jiaoyang Li**[3], **Erdem Bıyık**[1], **Sven Koenig**[4]
[1]Thomas Lord Department of Computer Science, University of Southern California
[2]Independent Researcher
[3]Carnegie Mellon University
[4]University of California, Irvine
{yimintan, biyik}@usc.edu, xiaoxiong.xx21@gmail.com, flotherxi@gmail.com, jiaoyanl@andrew.cmu.edu, sven.koenig@uci.edu

Introduction

Multi-Agent Path Finding (MAPF) is an NP-hard problem which focuses on finding collision-free paths for multiple agents in a known environment while optimizing a specified cost function. Many algorithms have been proposed to solve this problem or its variants, such as Conflict-Based Search (CBS), M^*, LaCAM and MAPF-LNS2. Recently, learning-based MAPF solvers have garnered significant attention (Alkazzi and Okumura 2024). Currently, all learning-based MAPF solvers adopt decentralized approaches, where each agent takes surrounding local information as input, typically represented as a field-of-view (FOV). Many decentralized methods have been proposed, such as PRIMAL and MAPF-GPT. As features are based on the agent itself, these approaches inherently allow the number of agents to vary. On the other hand, centralized approaches bring several benefits, such as the ability to coordinate the movements of multiple agents. However, the literature lacks centralized MAPF algorithms that are *learning-based*, since it is challenging to train a centralized neural network that can handle variability in both number of agents and map sizes.

We present the first centralized learning-based method RAILGUN for MAPF. The core idea of RAILGUN is to generate a directed graph in which each node has exactly one outgoing edge at every timestep. This design enables our method to handle any number of agents on the map. Additionally, we use CNN as the model backbone which produces outputs of the same dimensions as the input features. This allows RAILGUN to accommodate maps of varying sizes. In summary, our contributions are as follows: (1) We propose the first centralized learning-based MAPF algorithm, RAILGUN, which generates actions for map grid cells rather than for individual agents. (2) We design a CNN-based network enabling RAILGUN to handle maps of different sizes. (3) Through experiments in diverse test settings, we demonstrate that RAILGUN generalizes effectively to new types of maps and testing scenarios, and outperforms most baseline methods in POGEMA benchmark.

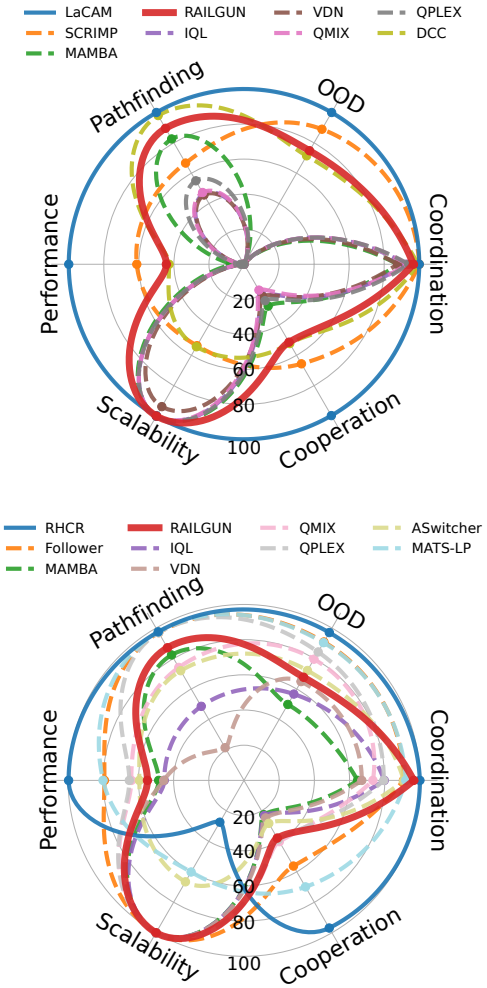

Figure 1: POGEMA Test (Top: MAPF, Bottom: LMAPF): Performance, OOD, Pathfinding, and Cooperation represent solution quality in terms of SoC/throughput. Scalability reflects runtime with respect to the number of agents. Coordination measures the probability of invalid actions.

[*]Equal contribution

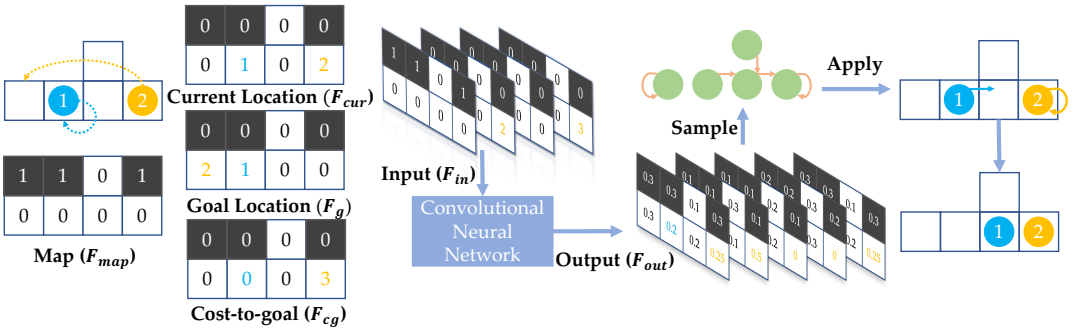

Figure 2: RAILGUN Inference Overview: On the left side, there is one current state along with all related input features of size $(n, m, 1)$. These features are then stacked along the last channel to construct the input feature F_{in} of size (n, m, k). In this example, we have $n = 2$, $m = 4$, and $k = 4$. On the right side, the input feature F_{in} is fed into a CNN-based neural network, which outputs action probabilities F_{out} of size $(n, m, 5)$.

Method

First, we discuss why it is difficult to design a learning algorithm for centralized MAPF where policies are agent-based. When focusing on generating actions based on agent features, we need to provide neural network with at least agent's start location, goal location, and additional features, amounting to k scalar variables ($k \geq 4$) for one agent. Then the total number of features is at least kN. Consider the maximum number of agents could be $N = |V| \approx nm$, where n and m are the 2D map dimensions. If we want to handle all possible numbers of agents on a specific map, the total feature size would be knm. This dependence on map size means we cannot create a policy to cover all different maps if we construct the features agent by agent. That is why all learning-based MAPF solvers adopt a decentralized approach with a limited FOV for each agent (Alkazzi and Okumura 2024).

Our insight is that in a valid MAPF solution, there will be no collision, which means there can be at most one agent in each map grid cell in each timestep. At any timestep, each agent chooses one of the five edges of its grid cell as its action. Therefore, if we remove all edges that the agents do not use at each timestep, we find that a valid MAPF solution can be viewed as a series of specialized graphs. As shown in Figure 2, these specialized graphs have exactly one edge in every occupied grid cell. Once such a directed graph is given, no MAPF solver is needed, as there is only one possible transition at each timestep. The sequence of these specialized graphs then constitutes a valid MAPF solution.

After converting the agent-based solution into a series of specialized graphs, we use a CNN network to address the challenge of generating these specialized graphs and generalizing across different maps, which we discussed in the previous paragraphs. In this paper, we use standard U-Net architecture for RAILGUN, where the input feature is F_{in} with size (n, m, k) and the output feature is F_{out} with size $(n, m, 5)$. Here, k represents the number of feature channels based on the feature design, and (n, m) represents map size. The resulting RAILGUN model contains approximately 30 million FP32 parameters. We show an example in Figure 2. We use POGEMA (Skrynnik et al. 2025) to test our method. POGEMA includes several different metrics, allowing a fair multi-fold comparison. For data collection, our training data is generated by LaCAM-v1 (Okumura 2023).

Experiments & Results

The model is trained with cross-entropy loss and a batch size of 256. The training process achieves convergence in only six hours, leveraging the power of four NVIDIA A100 GPUs. For training data, we randomly generate 180 maze maps with 32×32 size, each with varying obstacle densities and maze shape and a total of 127 scenarios for each map. All experiments were conducted on a system running with an AMD Intel i9-12900K CPU and NVIDIA RTX 3080. For the testing phase, POGEMA provides a total of 3,376 test cases, varying numbers of agents, and different map sizes.

In top of Figure 1, when testing on MAPF, we observe that RAILGUN achieves high scores across all six metrics compared to other learning-based methods. However, it still exhibits a significant gap with LaCAM in SoC-related metrics. This outcome is expected, as RAILGUN is trained on data generated by LaCAM-v1, and LaCAM-v1 is not designed to achieve the best SoC performance. Mimicking LaCAM-v1 is the top priority of RAILGUN rather than producing a valid solution with the lowest SoC. As shown in bottom of Figure 1, when testing on LMAPF, a completely zero-shot task for RAILGUN, RAILGUN achieves only moderate scores in throughput-related metrics since none of the training data was optimized for throughput. However, this zero-shot test also demonstrates RAILGUN's strong generalization ability across different tasks.

References

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

Okumura, K. 2023. Lacam: Search-based algorithm for quick multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

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 Representations (ICLR)*.