Learned Representations Enhance Multi Agent Path Planning

Marius Captari [1] **Herke van Hoof** [1]

Abstract

Multi-Agent Pathfinding (MAPF) involves coordinating multiple agents to find collision-free paths in a shared environment. For large-scale instances, sub-optimal heuristics can be used that are either hand-crafted or learned from data. In this paper, we attempt to combine these approaches by training a neural network to modify problem representations such that Prioritized Planning, a conventional heuristic solver, will produce closer-to-optimal solutions. Thereby, we can leverage the strong performance of existing heuristics with the flexibility of data-driven algorithms. Training the neural network requires propagating learning signals through prioritized planning. This is achieved by calculating gradients of a relaxation of the algorithm using a black-box differentiation approach. Experiments on standard MAPF benchmarks demonstrate that our approach reduces PP's optimality gap without significantly compromising computational efficiency.

1. Introduction

Integrating planning algorithms with learning-based methods offers a promising strategy for addressing complex decision-making challenges. Classical planners such as constraint solvers, graph search algorithms, and symbolic planners, offer strong theoretical guarantees, interpretability, and robustness derived from their structured, rule-based representations. However, these methods often require detailed problem formulations, potentially limiting their adaptability in dynamic environments and fail to scale to more realistic scenarios. On the other hand, learning-based methods have demonstrated remarkable flexibility and generalization capabilities by extracting latent structures directly from data (Silver et al., 2016). Yet, they commonly lack the interpretability, compositionality, and robustness offered by classical symbolic or programmatic representations, making their decisions difficult to verify or safely deploy in real-world scenarios.

Motivated by these complementary strengths, recent work explores hybrid planning–learning systems (Bengio et al., 2021). Rather than replacing symbolic solvers, these methods inject learning into targeted components such as heuristics, reward shaping, program synthesis, or symbolic cost function, to blend structured reasoning with data-driven flexibility.

In this work, we focus on Multi-Agent Pathfinding (MAPF), a sequential decision-making task that requires coordinating multiple agents on a shared graph to reach goals without collisions. MAPF lies at the core of many real-world systems, ranging from automated warehouse fleets and airport ground vehicle coordination to drone swarms and multi-robot exploration—where efficient, collision-free routing translates directly into higher throughput, lower energy consumption, and improved safety. Classical MAPF methods offer clear guarantees but scale poorly with increasing complexity, whereas heuristic-based planners scale better but yield suboptimal solutions. Previous efforts to enhance MAPF solvers through learning have predominantly modified local planner decisions, such as agent prioritization (Zhang et al., 2022) or conflict resolution (Huang et al., 2021). However, these local modifications do not fully leverage the global structure of the underlying representation, potentially limiting solution quality.

To address this, we propose a global, differentiable representation learning method: we train a neural network to adjust graph edge weights such that a fast, heuristic planner, Prioritized Planning (PP) (Silver, 2005), produces solutions closer to optimality. Our learned graph representation remains structured and interpretable, aligning with the broader goal of programmatic representation. We use black-box differentiation (Pogančić et al., 2019) to propagate gradients through the non-differentiable PP algorithm, enabling end-to-end learning of the graph representation without compromising solver efficiency.

Our primary contributions are twofold. First, we introduce a differentiable representation-learning framework that glob-

[1]University of Amsterdam. Correspondence to: Marius Captari <m.captari@uva.nl>.

ICML 2025 Workshop on Programmatic Representations for Agent Learning, Vancouver, Canada.

ally reshapes MAPF instances by learning edge-cost adjustments rather than focusing on local planner heuristics or priority decisions. Second, we demonstrate empirically that this approach shrinks the optimality gap of PP while preserving its computational efficiency.

2. Related Work

Integrating planning with machine learning often involves embedding symbolic solvers into neural architectures. For example, Value Iteration Networks (Tamar et al., 2016) insert a differentiable value-iteration module into CNNs to perform implicit planning, and Neural A* Search (Yonetani et al., 2021) learns heuristic functions for a differentiable A* algorithm—improving interpretability and generalization over reactive policies.

Predict-then-Optimize trains models to predict problem parameters (e.g., costs, demands) that are passed unchanged into classical solvers, with learning guided by downstream decision loss rather than raw accuracy (Elmachtoub & Grigas, 2022). In contrast, our method learns to alter the problem representation itself so that a fast, suboptimal planner produces higher-quality solutions.

To enable end-to-end gradient flow through inherently discrete solvers such as shortest-path, constraint, and combinatorial optimizers, researchers employ continuous relaxations, perturbation-based approximations, or black-box differentiation (Berthet et al., 2020; Pogančić et al., 2019), successfully integrating solvers into learning pipelines for ranking (Rolínek et al., 2020) and graph optimization (Karalias & Loukas, 2020) without fundamentally altering the solver itself.

Within Multi-Agent Path Finding (MAPF), most research integrating learning has focused on enhancing local solver components. For optimal solvers, such as Conflict-Based Search (CBS) (Sharon et al., 2015), learning-based methods have improved efficiency by guiding conflict resolution or node selection (Huang et al., 2021). For heuristic solvers like PP, neural models have been successfully used to learn effective agent prioritizations (Zhang et al., 2022). Similarly, Large Neighbourhood Search (LNS) approaches leverage learning to intelligently select subsets of agents for iterative replanning or to embed local sub-problems into learned architectures (Li et al., 2021a; Huang et al., 2022; Yan & Wu, 2024). Decentralized reinforcement learning methods, such as PRIMAL (Sartoretti et al., 2019), further scale to larger scenarios but often compromise optimality and completeness guarantees for scalability.

Despite these advancements, existing learning-augmented MAPF approaches predominantly target local decision components such as heuristics or priority ordering, rather than altering the underlying global representation of the planning problem itself. To our knowledge, no prior work has leveraged black-box differentiable optimization techniques to globally reshape graph-based representations explicitly to guide heuristic MAPF planners towards more optimal solutions.

3. Method

Our work aims to bridge this gap by applying differentiable optimization to learn structured graph representations, thus globally influencing heuristic solvers to enhance the quality of their solutions without sacrificing computational efficiency.

We propose a differentiable learning framework for guiding PP toward near-optimal solutions by modifying the edge weights of the planning graph. The proposed system is illustrated in Figure 1. A neural network predicts instance-specific edge costs, which are used by PP to generate plans. The network is trained to minimize the deviation from optimal solutions produced by EECBS (Li et al., 2021b), combining PP's efficiency with data-driven adaptability. Since the mapping from edge weights to solver output is piecewise constant, standard backpropagation yields zero gradients. To address this, we apply the black-box differentiation technique from Pogančić et al. (2019), which enables end-to-end training by approximating informative gradients via perturbed planner evaluations. The remainder of this section formalizes the MAPF setting and PP, introduces the surrogate gradient, describes the neural cost model, and outlines the training procedure.

3.1. MAPF Problem Definition

A MAPF instance is defined on an undirected graph $G = (V, E)$ with $|V| = N$ vertices (grid cells) and $|E| = M$ edges between neighbouring cells. For every vertex $v \in V$, we also include a self-loop $\{v, v\} \in E$ to explicitly encode wait actions.

We consider a set of n agents $\mathcal{A} = \{a_1, \ldots, a_n\}$, each with a start vertex $s_i \in V$ and a goal vertex $g_i \in V$. Each agent moves over discrete time steps. Let $w \in \mathbb{R}^M_{\geq 0}$ denote a non-negative vector of edge costs, indexed according to the edges in E.

A joint plan is *feasible* if it avoids any *vertex conflicts*—two agents occupying the same vertex at the same timestep—and *edge conflicts*—two agents traversing the same undirected edge in opposite directions simultaneously.

We further define the edge-usage vector $y \in \mathbb{N}^M$, where

$$y_e = \sum_{i,t} \mathbf{1}\big[(v_i^t, v_i^{t+1}) = e\big] \tag{1}$$

counts how many times edge e is traversed (including self-

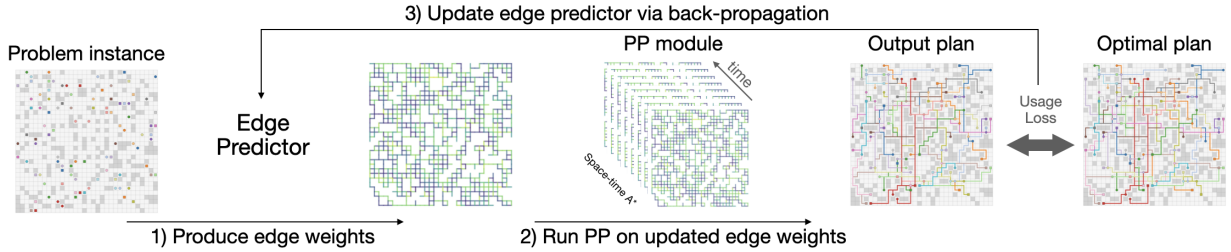

Figure 1. Differentiable MAPF training framework that learns edge-cost adjustments via black-box gradients from expert plan comparisons.

loops for waits). Given the edge costs w, the total plan cost is the sum of each edge's cost multiplied by its usage

$$c(w, y) \;=\; \sum_{e \in E} w_e \, y_e. \tag{2}$$

Let $\mathcal{Y}$ be the set of all feasible edge-usage vectors (i.e. collision-free plans). Then the planner solves the following discrete optimization problem:

$$y^\star(w) \;=\; \arg\min_{y \in \mathcal{Y}} c(w, y). \tag{3}$$

Prioritized planning solves this problem heuristically by assigning a fixed priority order to agents and planning their paths sequentially. Each agent computes its shortest path using A* on a space–time graph, treating reserved vertices and edges from higher-priority agents as dynamic obstacles (Silver, 2005). This defines a deterministic mapping from edge costs w to a feasible joint plan $y(w)$. While PP is fast and scales well with the number of agents, its greedy structure often results in suboptimal global solutions. The goal of this work is to learn cost vectors w such that $y(w)$ more closely approximates a globally optimal plan and thus minimizes the total sum-of-costs.

3.2. Black-Box Differentiation through PP

Intuitively we can think of PP as a mapping $w \mapsto y(w)$ which is piecewise constant, yielding zero gradients almost everywhere. To obtain meaningful learning signals, we apply the continuous perturbation method of Pogančić et al. (2019). Specifically, we define the task loss $L(\hat{y}, y^\star)$ as the mean squared error between the predicted ($\hat{y}$) and optimal ($y^\star$) *directed* edge-usage vectors:

$$L(\hat{y}, y^\star) = \frac{1}{M} \sum_{e \in E} (\hat{y}_e - y^\star_e)^2, \tag{4}$$

where each directed edge $e = (u, v)$ is treated distinctly from its reverse edge (v, u). Given a scalar $\lambda > 0$, a perturbed cost vector is constructed as:

$$w' = w + \lambda \frac{\partial L}{\partial \hat{y}}. \tag{5}$$

Evaluating PP with w' yields a perturbed solution $y_\lambda = y(w')$, from which we compute the surrogate gradient:

$$\nabla_w f_\lambda(w) = -\frac{1}{\lambda}(\hat{y} - y_\lambda). \tag{6}$$

This approximation enables backpropagation through the planner with exactly two calls to PP and without modifying its internals.

3.3. Neural Cost Shaping

We now introduce a neural network $\mathcal{N}_\theta$ which maps instance features x (the map, static obstacles, and agent start/goal pairs) into a vector of edge costs $w = \mathcal{N}_\theta(x)$.

Gradients obtained via (6) are propagated through $\mathcal{N}_\theta$ to update the parameters θ. Our aim is that this process encourages the network to inflate costs along edges that lead to downstream conflicts and to discount those that promote globally efficient paths.

During planning, the predicted edge weights w are used as cost values in the single-agent A* searches performed by PP. However, to preserve the validity of the reservation table which encodes blocked edges and vertices over time we retain the original graph costs to determine traversal durations. We employ a *true distance* heuristic in A*, computed as the shortest-path distance from each agent's start to its goal on the obstacle-free graph: this heuristic is admissible (it never overestimates the true cost). During training, we recompute this heuristic under the current learned weights to ensure it remains admissible relative to the modified cost function. This maintains consistency with the underlying environment dynamics while allowing the learned weights to influence the planner's route preferences.

3.4. Training

Training is performed on *scenarios*—sets of n agent start–goal pairs defined on the same map as outlined in Algorithm 1. Each mini-batch consists of B independent scenarios. For each scenario, edge costs are predicted, the plan is computed via PP, a perturbed plan is obtained, and the surrogate gradient is applied.

Algorithm 1 One training epoch with differentiable PP

Input: Mini-batch $\{x_j, y_j^\star\}_{j=1}^B$, smoothing parameter λ
Output: Updated network parameters θ
for $j = 1$ **to** B **do**
  $w_j \leftarrow \mathcal{N}_\theta(x_j)$ {predict edge costs}
  $\hat{y}_j \leftarrow \mathrm{PP}(w_j)$ {forward pass}
  $w'_j \leftarrow w_j + \lambda\, \partial L / \partial \hat{y}_j$
  $y_{\lambda,j} \leftarrow \mathrm{PP}(w'_j)$ {perturbed pass}
  $\nabla_{w_j} \leftarrow -(\hat{y}_j - y_{\lambda,j})/\lambda$
end for
Back-propagate $\{\nabla_{w_j}\}$ and update θ

The model is trained for T epochs using randomly sampled training scenarios. Evaluation is performed on held-out start/goal configurations drawn from the same map. Generalization to new maps is left for future work.

4. Experiments

We evaluate our method on scenarios from the `random-32-32-20` map, which is part of a standard benchmark from the MAPF definitions and variants suite (Stern et al., 2019). We consider instances with $n \in \{50, 75, 100\}$ agents to assess performance across varying levels of congestion and complexity.

Training Setup. We use Enhanced Edge-Conflict Based Search (EECBS) (Li et al., 2021b), a CBS variant with edge constraints and focal search, to generate expert plans and their edge-usage vectors $\hat{y}$ for loss computation. We employ a sub optimality bound of 1.0 for 50-agent scenarios and 1.05 for more challenging ones, trading off optimality and runtime. On a 36-core machine, we were able to generate solutions for all 500 training scenarios within a total time budget of approximately 1 hour.

We use 500 training and 25 test scenarios (20 % of training for validation). The edge predictor is trained with our earlier loss, matching PP's directed edge-usage to EECBS, starting from uniform costs of 1. Agents are ordered by ascending true distance so shorter-path agents plan first; if PP fails, we fix a single random permutation. The same ordering is used throughout training (for $\hat{y}$) and testing for a fair comparison.

Model Architectures and Optimization. We compare two variants of our differentiable MAPF framework. The **Edge** model maintains a learnable scalar cost for each graph edge (initialized to the original unit weight values of 1.0) and ignores start/goal inputs, directly predicting the full edge-weight vector. The **GNN** model instead computes edge weights via a two-layer GraphSAGE network (Hamilton et al., 2017) over the grid: node features encode a simple demand signal (starts vs. goals), sinusoidal positional embeddings, and a small agent ID embedding; after message passing, each edge's weight is produced by a lightweight MLP over its two endpoint embeddings. Both models are trained end-to-end using the described loss with Adam (learning rate $5e-4$) and a $\lambda = 1.0$.

Evaluation. Model selection uses validation delay (PP's sum-of-costs minus the collision-free lower bound). The model checkpoint that achieves the lowest validation delay is selected as the final model. The selected model is then evaluated on 25 held-out test scenarios, reporting delay and optimality gap relative to EECBS.

Table 1. Average delay and optimality gap (relative to EECBS) on 25 unseen test scenarios. Lower is better.

Map	Agents	Average Delay (% gap)			EECBS
		PP (orig.)	PP (Edge)	PP (GNN)	
random	50	56.20 (134.6%)	52.96 (121.0%)	**51.36 (114.3%)**	23.96
	75	163.08 (141.8%)	151.68 (124.9%)	**149.64 (121.9%)**	67.44
	100	354.20 (118.9%)	333.44 (106.1%)	**317.12 (95.9%)**	161.80

Discussion. Experiments on the `random-32-32-20` map (Table 1) demonstrate that learning a structured graph representation can improve the solution quality of a fast heuristic planner without sacrificing its efficiency. Across all agent counts $(50, 75, 100)$, both learned models—direct per-edge parameters and the GNN-based predictor—consistently reduce the average delay of PP relative to the original uniform-cost baseline. The GNN model yields the largest gains highlighting the value of conditioning edge costs on global context and agent interactions.

Importantly, these benefits incur minimal overhead. PP alone takes 0.0183s versus 0.0263s with our learned model (+0.0080s), since learned weights don't alter PP's inner loop, preserving scalability for real-time systems.

5. Conclusion

In this work, we combine black-box differentiable planning with learned edge-cost shaping to enhance an existing MAPF heuristic solver, offering a practical bridge between structured planning and data-driven adaptability.

While our experiments demonstrate gains on a single map topology using EECBS demonstrations, future extensions should evaluate the framework on diverse graph structures to assess generalization and develop expert-free approaches such as reinforcement learning to reduce reliance on costly expert demonstrations.

Acknowledgements

This research project is part of the AI4REALNET project. AI4REALNET has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

Elmachtoub, A. N. and Grigas, P. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Huang, T., Koenig, S., and Dilkina, B. Learning to resolve conflicts for multi-agent path finding with conflict-based search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11246–11253, 2021.

Huang, T., Li, J., Koenig, S., and Dilkina, B. Anytime multi-agent path finding via machine learning-guided large neighborhood search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9368–9376, 2022.

Karalias, N. and Loukas, A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

Li, J., Chen, Z., Harabor, D., Stuckey, P. J., and Koenig, S. Anytime multi-agent path finding via large neighborhood search. In *International Joint Conference on Artificial Intelligence 2021*, pp. 4127–4135. Association for the Advancement of Artificial Intelligence (AAAI), 2021a.

Li, J., Ruml, W., and Koenig, S. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 12353–12362, 2021b.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., and Rolinek, M. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.

Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C., and Martius, G. Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630, 2020.

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

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

Silver, D. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pp. 117–122, 2005.

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. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, pp. 151–158, 2019.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

Yan, Z. and Wu, C. Neural neighborhood search for multi-agent path finding. In *The Twelfth International Conference on Learning Representations*, 2024.

Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M., and Kanezaki, A. Path planning using neural a* search. In *International conference on machine learning*, pp. 12029–12039. PMLR, 2021.

Zhang, S., Li, J., Huang, T., Koenig, S., and Dilkina, B. Learning a priority ordering for prioritized planning in multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 15, pp. 208–216, 2022.