Comparative Analysis of Data-Driven GNN-Based Path Planning and Classical Search Algorithms

Navneet Kishan Srinivasan
Muhammad Shaizal Yasin
Isroilbek Jamolov
Md. Rakibul Hasan

Friedrich-Alexander-Universität (FAU)
Introduction to Control and Machine Learning

February 2nd, 2026

Outline

The Path Planning Dilemma

Optimality vs. Scalability

The Goal: Find a collision-free path π^* from S to G.

The Classical Trade-off:

- **Search Algorithms (A*):**
  - Pros: Guarantees optimality (shortest path).
  - Cons: Computationally expensive. Cost $\propto$ Nodes Expanded.
- **Heuristics:**
  - Performance depends entirely on the heuristic $h(n)$.
  - Calculating complex heuristics is slow.

The Research Question

Can we replace this expensive search process with a learned 'intuition'? Can a GNN look at a map and directly output the optimal path?

Problem Formulation

Defining the Search Space

We model the environment as a grid graph $G = (V, E)$.

- **Nodes (V):** Free cells in an $N \times N$ grid ($N = 20$).
- **Edges (E):** Defined by 4-connectivity:

$$(u, v) \in E \iff \|u - v\|_1 = 1$$

- **Objective:** Find a sequence of nodes $\pi = (v_1, \dots, v_T)$ such that:

$$\pi^\star = \arg\min_{\pi \in \Pi} \sum_{t=1}^{T-1} c(v_t, v_{t+1})$$

In our uniform-cost setting, $c(\cdot) = 1$, minimizing **hop count**.

BFS & Dijkstra

Uninformed Search

Before applying Deep Learning, we establish classical baselines.

1. **Breadth-First Search (BFS)**
   - Explores in "layers" (ripples).
   - **Optimality:** Guaranteed for unweighted graphs.
   - **Drawback:** Expands in all directions equally.

$$O(|V| + |E|)$$

2. **Dijkstra's Algorithm**
   - Generalized BFS for weighted graphs.
   - Relaxes edges: $d(v) = \min(d(v), d(u) + w_{uv})$.
   - **In our grid:** Reduces to BFS since $w = 1$.

A* Search

Informed Search

A* directs the search towards the goal using a heuristic function.

The Cost Function

$$f(n) = \underbrace{g(n)}_{\text{Cost-to-come}} + \underbrace{h(n)}_{\text{Heuristic (Estimated cost-to-go)}}$$

Implementation Details:

- We use **Manhattan Distance** as the admissible heuristic:

$$h(n) = |n_x - goal_x| + |n_y - goal_y|$$

- **Key Property:** Since $h(n) \leq h^*(n)$, A* guarantees the optimal path.

This algorithm serves as the "Teacher" to generate ground-truth labels for our GNN.

Learning Formulation

What our model predicts

We learn a per-node predictor:

$$\hat{y}_i \approx P(v_i \in \pi^\star \mid G, s, g).$$

Define node features $\boldsymbol{x}_i \in \mathbb{R}^8$:

$$\boldsymbol{x}_i = \left[\underbrace{\text{isBlocked}}_{1}, \underbrace{\text{isStart}}_{1}, \underbrace{\text{isGoal}}_{1}, \underbrace{x_{\text{norm}}, y_{\text{norm}}}_{2}, \underbrace{d_{\text{Man}}(i,g)_{\text{norm}}, d_{\text{Man}}(i,s)_{\text{norm}}}_{2}, \underbrace{d_{\text{Euc}}(i,g)_{\text{norm}}}_{1}\right].$$

Training signal: soft labels $y_i \in \{0, 0.3, 1\}$

Message Passing to Outputs

GraphSAGE $\rightarrow$ Node probabilities

Message Passing (GraphSAGE):

$$h_i^{(k+1)} = \sigma\left(W^{(k)} h_i^{(k)} + U^{(k)} \cdot \mathrm{AGG}\{h_j^{(k)} : j \in \mathcal{N}(i)\}\right)$$

Output Head:

$$z_i = \boldsymbol{w}^\top h_i^{(L)} + b, \qquad \hat{y}_i = \sigma(z_i)$$

Usage:

- **Greedy:** move to neighbor with max $\hat{y}$
- **Neural A*:** use $\hat{y}$ as a bias in A* scoring

Decoding Strategies

From node probabilities to paths

Greedy Decoding (GNN-only)	**Neural A* (GNN-guided)**
Move to neighbor with highest $\hat{y}$	Modified A* scoring function
$\boxed{v_{t+1} =_{u \in \mathcal{N}(v_t)} \hat{y}_u}$	$\boxed{f(n) = g(n) + h(n) + \alpha(1 - \hat{y}_n)}$
Fast but fragile Local maxima, no backtracking Low success rate	**Robust hybrid search** Preserves search structure High success rate

Data Generation Pipeline

From random grids to PyG graphs

Pipeline used:

1. Sample grid with obstacle probability $p_{\text{block}} = 0.2$.
2. Sample start/goal from free cells.
3. Run pure A* to get $\pi^\star$ (discard if no path).
4. Convert grid to PyG graph:

$$\text{Data}(x \in \mathbb{R}^{|V| \times 8},\ edge\_index \in \{1, \ldots, |V|\}^{2 \times |E|},\ y \in \mathbb{R}^{|V|}).$$

5. Apply soft labels (1.0/0.3/0.0).

Dataset: $\sim$1.2k valid samples **Split:** 70/15/15 (train/val/test)

Training Setup

Loss, imbalance handling, optimiser

Model: GraphSAGE ($L = 5$, hidden 128, dropout 0.2)

Base loss (free cells only):

$$\mathcal{L} = -\sum_{i \in \text{Free}} \Big(y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i)) \Big)$$

Imbalance-aware loss:

$$\mathcal{L}_{\text{pw}} = \text{BCEWithLogitsLoss(pos\_weight)}, \quad \text{pos\_weight} \approx \frac{\#\text{neg}}{\#\text{pos}} \approx 18$$

Optimiser: Adam, $\eta = 10^{-3}$

Evaluation Protocol

Thresholding and F1-based tuning

Thresholding:

$$\hat{y}_i^{(\text{hard})} = \nVdash[\sigma(z_i) > \tau]$$

Metrics: Precision, Recall, and F_1 (preferred due to sparse positives)

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$$

Threshold tuning (validation):

$$\tau^\star =_{\tau \in [0,1]} F_1(\tau)$$

Result:

$$\tau^\star \approx 0.85, \qquad F_1^{(\text{val})} \approx 0.43$$

Accuracy is not reliable here due to highly sparse path labels.

Planner-Level Evaluation

Compared planners and efficiency measures

Planners evaluated on identical test graphs:

- **Pure A***: Manhattan heuristic (baseline)
- **Greedy-GNN**: follow max-probability neighbor
- **Neural A***: A* with learned bias $\alpha(1-\hat{y})$, $\alpha = 2.0$

What we measure (planner-level):

- **Success rate**
- **Path length**
- **Runtime**
- **Node expansions**

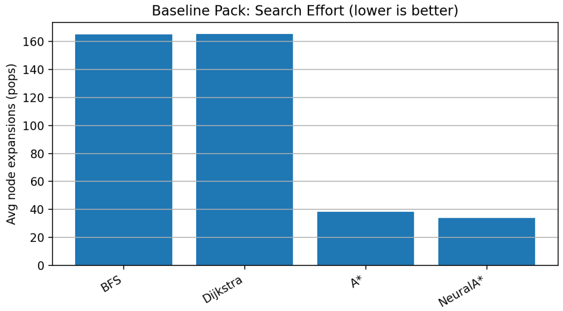
Why expansions: a more stable algorithmic signal than time, which includes Python/GPU overheads.

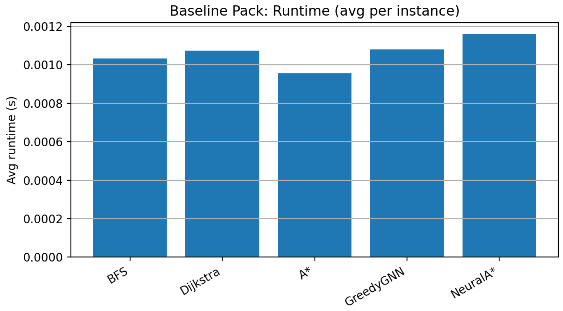
Baseline Pack: Protocol and Key Results

Efficiency (Expansions & Runtime)

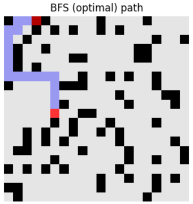
Protocol: Evaluate all planners on the *same* held-out test graphs.
Planners: BFS, Dijkstra, A*, Greedy-GNN, Neural A*.

Key outcome: *Greedy-GNN shows weaker end-to-end performance, while Neural A* aligns closely with classical search behavior in this setting.*

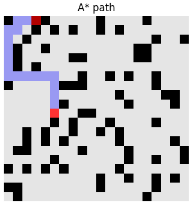




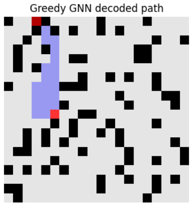
Qualitative: Same Map, Different Planners



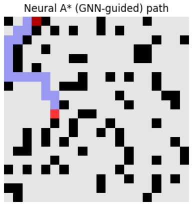
BFS (optimal)



A*



Greedy-GNN



Neural A* (GNN-guided)

Baseline Pack: Summary and Takeaways

Mean over test instances

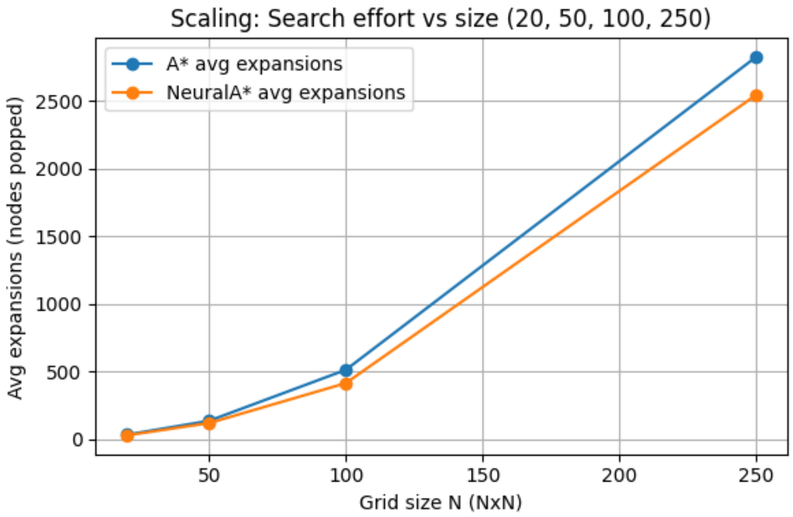
All methods: 100% success on this test set (solvable instances).

Method	**Len Ratio vs BFS**	**Avg Time (ms)**	**Avg Expansions**
BFS	1.000	0.619	156.32
Dijkstra	1.000	0.744	156.48
A*	1.000	0.677	33.58
Greedy-GNN	1.295	0.763	N/A
Neural A*	1.000	0.819	29.10

- **Search baselines:** BFS/Dijkstra/A* return shortest paths on unit-cost grids.
- **Greedy-GNN:** local decoding can fail (local maxima; no backtracking) and yields longer paths.
- **Neural A*:** matches A* reliability/length, with similar (often slightly lower) expansions.

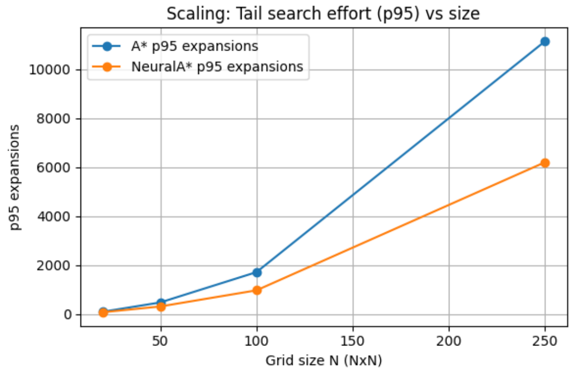
Note: Expansions are defined only for search-based planners (not Greedy-GNN).

Size Scaling Experiment Result



Robustness View: Tail Search Effort (p95 Expansions)

Hard cases



- **Tail effort:** Neural A* < A* (p95 expansions).
- **Scaling:** gap grows with N (bigger maps $\Rightarrow$ bigger benefit).

Conclusion and Limitations

What we learned and what remains

Main findings:

- **GNN signal:** learns meaningful path saliency, but node scores alone do not guarantee a valid path.
- **Decoding:** greedy is fragile; **Neural A*** keeps search reliability while using the GNN to cut search effort.
- **Scaling:** guidance transfers zero-shot from 20×20 to larger grids (solvable-only test sets).

Limitations:

- **No optimality guarantee:** learned bias can break admissibility.
- **Distribution dependence:** trained at $N{=}20$, $p_{\text{block}}{=}0.2$; performance may shift with size/density and limited samples.

Future Work and Related Work

Next steps:

- **Generalisation:** evaluate across more grid sizes and obstacle densities.
- **Learned heuristic:** predict cost-to-go and integrate it into A*.
- **Decoding:** replace greedy with beam search or constrained shortest-path over high-score nodes.
- **Scaling:** report expansions vs. grid size (optionally p95).

Related work: Zhou et al., arXiv:1812.08434 — GNNs as message passing + design pipeline; motivates robustness/generalisation questions.

Thank you!

Questions?

Slides on ResearchGate

You can find these slides here:

`https://www.researchgate.net/publication/400299517`

Presentation