AI-Guided A Path Planning with Predictive Search Corridors and Adaptive Expansion

Cristian-Marian Pascu

University of Wolverhampton
United Kingdom

This manuscript is based on coursework completed as part of the MSc Computer Science with Artificial Intelligence programme at the University of Wolverhampton and has been revised for public dissemination.

Abstract

This study investigates reducing unnecessary node expansion in A* path planning through machine-learning-guided search corridors. Although A* guarantees optimal paths with an admissible heuristic, its efficiency may degrade as graph size and search complexity increase. Unlike approaches that mainly learn heuristics or expansion policies, the proposed method predicts a reduced corridor before search begins. High-probability nodes restrict the search space explored by A*, while adaptive expansion releases blocked candidates when the initial corridor is incomplete. A certification stage compares the corridor result against global A* to verify correctness. The method was evaluated on small- and medium-scale generated geometric graphs using Random Forest, GraphSAGE, and an oracle corridor. On the medium dataset, GraphSAGE improved over A* in 23.6% of cases and matched it in 69.9%, with 6.5% uncertified cases. The oracle improved over A* in 55.1% of cases and matched the remainder, indicating that further node-expansion reductions are theoretically achievable.

Table of Contents

1. Problem Statement

Path planning is a fundamental problem in autonomous mobile robotics, where an agent must determine an efficient route between a start and a goal while minimizing path cost. A* is widely used because it combines accumulated path cost with heuristic guidance and can guarantee optimal solutions when the heuristic is admissible (Hart et al., 1968). However, as graph size and search complexity increase, A* may still expand nodes that do not belong to the final shortest path, creating unnecessary computational effort.

This project addresses the problem of reducing unnecessary node expansion while preserving the reliability of classical A*. The central research question is whether machine learning can predict a useful search corridor before pathfinding begins, allowing A* to search only a smaller subset of the graph. The key challenge is not shortest-path computation itself, but predicting a corridor that is compact enough to reduce search effort and complete enough to retain the nodes required for the optimal path.

The application context is autonomous mobile robot navigation, where planning may need to run repeatedly under computational and time constraints. Recent guided and corridor-constrained methods show that limiting the search region can reduce ineffective node expansion (Che et al., 2026), but such corridors are often defined using geometric or safety rules. This project instead investigates a data-driven corridor learned from shortest-path examples.

The resulting system combines machine learning with classical search: A* runs inside the predicted corridor, adaptive expansion releases blocked candidates when the corridor is incomplete, and certification checks the result against global A*. The scope is graph-level global path planning, rather than perception, localization, low-level control, or dynamic obstacle avoidance.

2. Literature Review

A* remains a key baseline in graph-based path planning because it combines accumulated path cost with heuristic guidance while preserving optimality under suitable heuristic conditions (Hart et al., 1968). Its main limitation is that optimality does not guarantee efficiency: search cost still depends on the number of expanded nodes. This has motivated recent work on learned heuristics, neural search guidance, and corridor-constrained planning.

One direction is to use machine learning to improve the heuristic used by A*. Pándy et al. (2022) introduced PHIL, a graph neural network approach for learning graph search heuristics from data. Their results show that learning can reduce explored nodes in A*-style search. However, the learned component still acts mainly inside the heuristic function, meaning the planner may continue to operate over the full graph. The present project differs by using machine learning before search begins, predicting a corridor that explicitly restricts the graph region explored by A*.

A second direction integrates neural networks more directly with search. Yonetani et al. (2021) proposed Neural A*, a differentiable version of A* combined with a neural encoder. This demonstrates that deep learning and classical search can be combined in a trainable planning architecture. However, Neural A* is mainly framed around grid-like inputs and differentiable search, which can make it less directly aligned with explicit graph-node corridor prediction. In contrast, this project keeps A* as an interpretable planner and

uses machine learning only to define a reduced search region, allowing the final path to be inspected and certified against global A*.

Graph neural networks are relevant because path planning is naturally graph-structured. GraphSAGE learns node representations by aggregating information from local neighborhoods, allowing it to generalize inductively rather than memorizing fixed node identities (Hamilton et al., 2017). This is important because the model should learn transferable structural patterns rather than memorize graph or node identities. However, a GNN predictor alone cannot guarantee that all necessary path nodes will be included. This limitation motivates the adaptive expansion and certification components of the proposed architecture.

Corridor-constrained planning is another related direction. Che et al. (2026) proposed GCHybridA*, which uses a Voronoi-guided safe corridor to restrict Hybrid A* search and reduce ineffective node expansion. This supports the idea that limiting the search region can improve planning efficiency. However, their corridor is constructed geometrically using Voronoi paths and safety constraints. The present project investigates a learned alternative, where the corridor is predicted from node probabilities before A* is applied.

Overall, the literature shows a shift toward hybrid planning systems that combine classical search with learned or constrained guidance. Existing ML approaches usually learn heuristics, expansion guidance, or differentiable planning maps, while corridor-based methods often rely on geometric construction. This project sits between these directions by learning a probability-based search corridor, applying A* inside it, and using adaptive expansion plus certification when the predicted corridor is incomplete.

3. Commercial Context and Existing Implementations

Commercial autonomous navigation systems are widely used in mobile robotics, warehouse automation, logistics, and manufacturing. In these systems, path planning is part of a wider navigation stack that includes mapping, localization, obstacle detection, task allocation, traffic management, and safety monitoring. The present project is therefore positioned not as a replacement for a full navigation system, but as a possible optimization layer for global path planning.

Autonomous mobile robots in logistics provide a clear example. Mobile Industrial Robots offers fleet management software for coordinating multiple robots across a facility, including traffic control and mission coordination (Mobile Industrial Robots, 2026). Such systems depend on efficient route planning because robot fleets may need to update routes repeatedly as tasks, robot availability, and congestion change. A graph-based planner that reduces unnecessary search effort could support faster planning decisions in these environments.

Warehouse robotics shows the same requirement. Amazon's Proteus robot is presented as an autonomous mobile robot designed to move packages while navigating safely around people and objects (Amazon, 2024). This reflects a wider shift from fixed-path automation toward robots operating in shared and dynamic spaces, where navigation must remain reliable and computationally efficient.

Open-source robotics systems also demonstrate the importance of reusable planning components. ROS 2 Navigation2 provides modules for planning, control, behavior management, recovery behaviors, and map-based navigation (Open Navigation LLC, 2026). The proposed ML-guided corridor approach is most relevant at the global planning level, where it could reduce the graph region explored before a path is selected.

However, real deployment would still require integration with local planning, obstacle handling, and runtime validation.

4. Architecture and Implementation

4.1 System Overview

The implementation is software-based and evaluated on generated graph data, so no physical robot hardware is included; the architecture targets the global planning layer that could later be integrated into a mobile robot navigation stack.

The proposed architecture combines machine learning with classical A* search. Instead of generating the final path directly, the model predicts a reduced set of nodes likely to contain the shortest path. A* then searches inside this corridor. If the corridor is incomplete, adaptive expansion adds blocked candidate nodes and reruns the search. Finally, the result is certified against global A* to verify optimality.

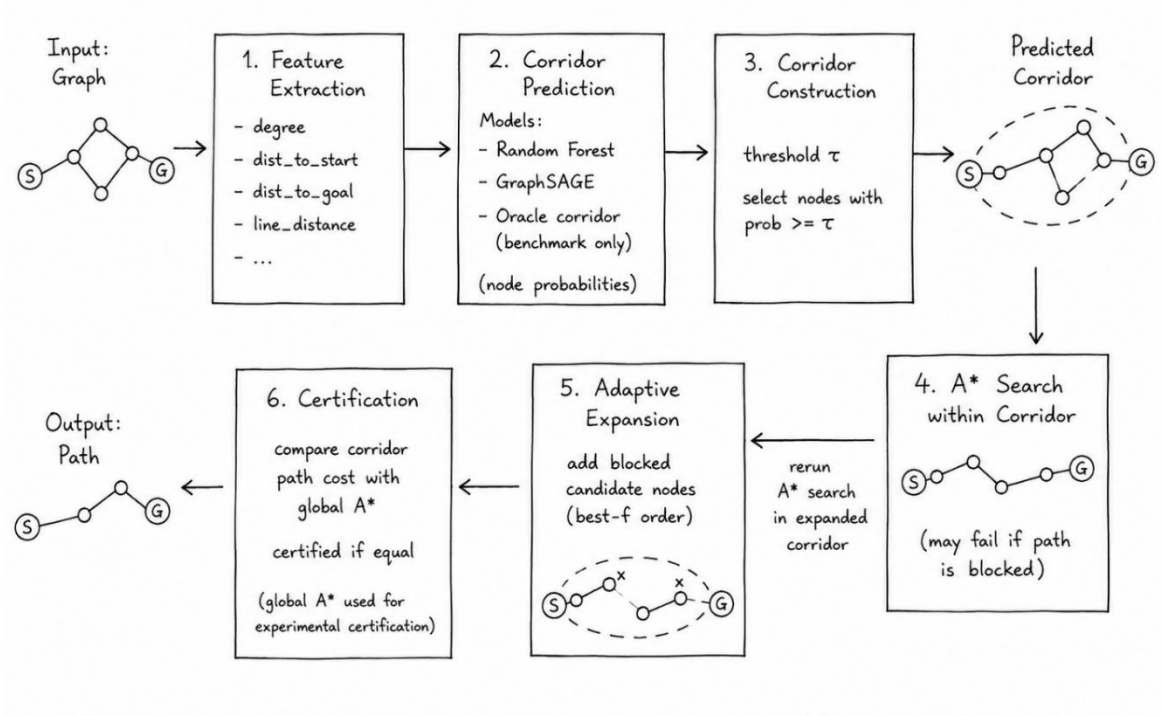

Figure 1. System overview of the ML-guided corridor search architecture. Node features are used to predict corridor probabilities, a threshold selects the initial corridor, A searches within it, adaptive expansion supports recovery when the corridor is incomplete, and certification verifies correctness.*

4.2 Dataset Generation and Graph Representation

The experimental data were generated as synthetic geometric graphs, with nodes represented by two-dimensional coordinates and weighted edges connecting nearby nodes. Edge weights were based on geometric distance and represented traversal cost, so each graph acted as an abstract map for global path planning.

For each graph, multiple start-goal queries were generated. The optimal path for each query was used to label nodes as path or non-path nodes, converting the problem into a node-level binary prediction task. The model therefore learns whether a node is likely to belong to the shortest path for a given start-goal pair.

Two datasets were used: a small dataset with 10–50 nodes per graph and a medium dataset with 80–200 nodes per graph. The medium dataset contained 738,950 node records and 73,893 edge records, with a positive label ratio of approximately 13.1%. Graph IDs, node IDs, start IDs, and goal IDs were excluded so that the models learned reusable structural and geometric patterns rather than memorizing specific graph instances.

4.3 Feature Engineering

For each node v in a start-goal query (s, g), a set of structural and geometric features was extracted to describe the node's local connectivity and its relationship to the planned route.

The most important engineered feature is the **line-distance**, defined as the perpendicular distance from node v to the straight-line segment connecting the start node s and goal node g:

$$d_{line}(v) = \frac{\|(v - s) \times (g - s)\|_2}{\|g - s\|_2}$$

where s, v, and g represent the 2D positions of the start, current, and goal node, respectively. This feature measures how far a node deviates from the direct start-goal direction. Nodes located close to the direct start-goal direction are expected to have smaller line-distance values, while nodes far from the route tend to have larger values.

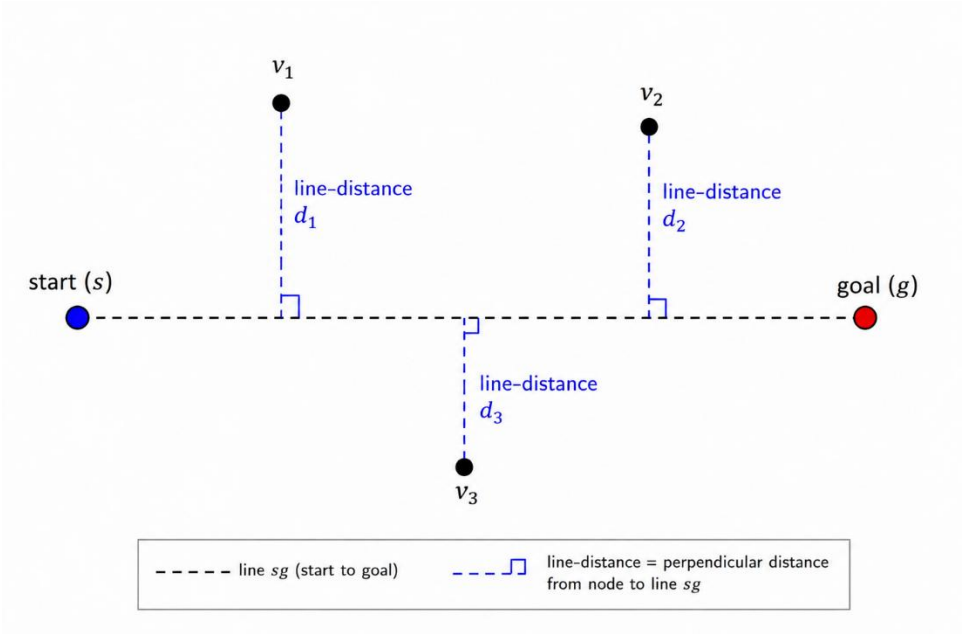

Figure 2. Illustration of the line-distance feature. For each node, line-distance is the perpendicular distance from the node to the straight start-goal line segment.

The complete feature set consists of node coordinates (x, y), node degree, distance to the start node $d(v, s)$, distance to the goal node $d(v, g)$, straight-line start-goal distance $d(s, g)$, average incident edge weight $\bar{w}(v)$, and line-distance $d_{line}(v)$.

In the medium dataset, non-path nodes have a mean line-distance of approximately 31.43, while shortest-path nodes have a mean line-distance of approximately 6.56, indicating that nodes located near the direct start-goal direction are substantially more likely to belong to the optimal path. Random Forest feature-importance analysis also identified line-distance as the most influential predictor, further supporting its usefulness for corridor prediction.

4.4 Corridor Prediction Models

4.4.1 Random Forest

Random Forest was implemented as a tabular corridor predictor using the engineered structural and geometric features. Both baseline and balanced versions were tested to address the class imbalance between path and non-path nodes. The model was useful as an interpretable baseline because feature importance confirmed the value of line-distance. However, it treats nodes mainly as independent feature vectors, with graph structure represented only indirectly through features such as degree and average edge weight.

4.4.2 GraphSAGE

GraphSAGE was implemented as the main graph-aware corridor predictor. Unlike Random Forest, it uses neighborhood aggregation, allowing node predictions to depend on both node attributes and surrounding graph structure. This is important for path planning because a node's usefulness often depends on how it connects to nearby nodes, not only on its individual features.

GraphSAGE outputs a probability for each node, and nodes above the selected threshold are included in the initial corridor. Two configurations were evaluated: a compact version, designed to prioritize search-space reduction, and a robust version using stronger adaptive recovery settings (`max_expansions=11`, `release_k=5`) to prioritise certification and reduce failed cases.

4.4.3 Oracle Corridor

An oracle corridor was included as an experimental upper bound rather than a deployable model. It uses knowledge of the true shortest path to estimate how much benefit corridor-guided search could theoretically provide if corridor prediction were perfect. Comparing Random Forest and GraphSAGE against the oracle helps separate limitations of the corridor-search strategy from limitations caused by imperfect prediction.

4.5 Corridor-Guided A* Search

After the initial corridor is constructed, A* is executed on the restricted subgraph rather than on the full graph. The standard A* priority function is preserved:

$f(n) = g(n) + h(n)$

where $g(n)$ is the accumulated cost from the start node and $h(n)$ is the geometric heuristic estimate to the goal. The machine learning model does not replace this heuristic; it only determines which nodes are initially available for expansion.

During corridor-guided search, A* expands only nodes inside the predicted corridor. If the corridor contains the necessary path nodes, A* can return the same optimal path while visiting fewer nodes than global A*. If the corridor is too large, the result may be a tie; if it is too narrow, important nodes may be missing and adaptive expansion is required. This keeps path construction deterministic while using learning only to restrict the search region.

4.6 Adaptive Expansion and Certification

The main risk of corridor-guided search is that the predicted corridor may exclude a node required for the optimal path. To recover from this, neighboring nodes outside the corridor are stored as blocked candidates rather than discarded. If A* cannot complete the search inside the current corridor, the best blocked candidates are released into the corridor and A* is rerun.

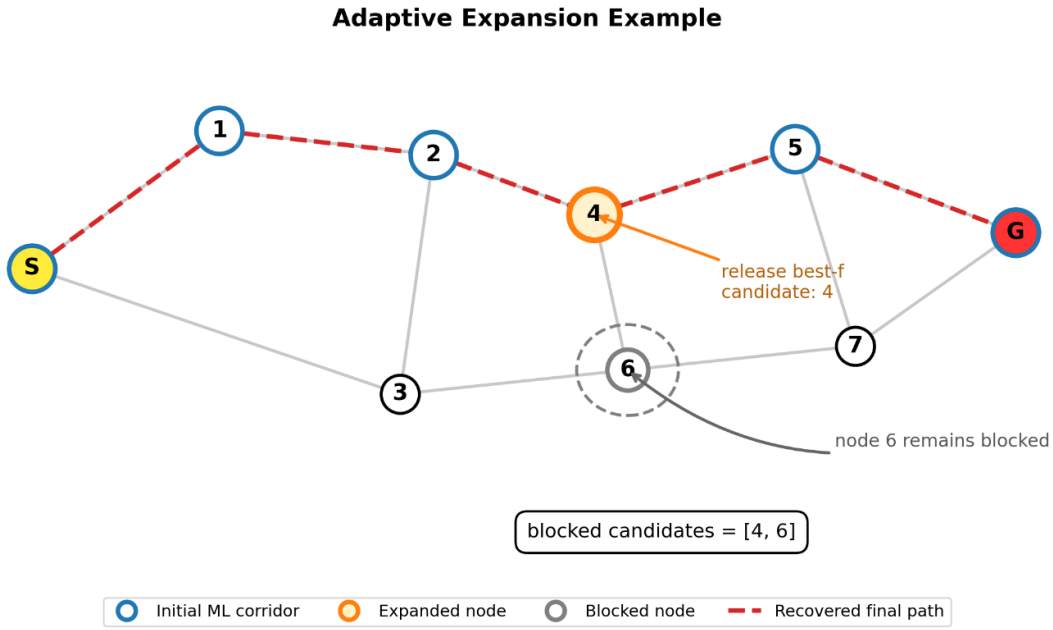

Figure 3. Simplified example of adaptive expansion.

The initial ML corridor contains the blue-outlined nodes. Nodes 4 and 6 are stored as blocked candidates, but node 4 is released as the best candidate and added to the corridor, allowing A* to recover the final path.

The expansion behavior is controlled by `release_k` and `max_expansions`. `release_k` defines how many blocked nodes are added per expansion step, while `max_expansions` limits the number of recovery attempts. This creates an efficiency-robustness trade-off: small releases keep the corridor compact, while larger releases improve recovery but can increase total visited nodes.

Certification verifies correctness during evaluation. The corridor-guided path cost is compared with the global A* path cost. If the costs match, the case is recorded as certified and optimal; otherwise, it is treated as uncertified. In this prototype, global A* acts as an experimental certifier. In a deployed system, a lighter fallback or validation mechanism would be required.

5. Evaluation and Discussion

5.1 Evaluation Setup and Metrics

Runtime was tested during experimentation, including Random Forest runs without certification, but it was not used as the main metric. On the generated graphs, global A* was already very fast, so runtime differences were small and strongly affected by prototype overhead such as Python execution, model inference, corridor construction, and repeated A* calls during expansion.

The primary metric was therefore the number of visited nodes, which more directly measures search-space reduction. Each case was classified as ML better, tie, A* better, or failed/uncertified depending on whether corridor-guided A* visited fewer, equal, or more nodes than global A*, or failed to certify the optimal path within the expansion limit.

5.2 Overall Model Comparison

Table 1. Positive-class node prediction performance for corridor predictors.

Model	**Precision**	**Recall**	**F1-score**
RF baseline	0.920	0.530	0.680
RF balanced grid	0.478	0.880	0.620
GraphSAGE	0.574	0.921	0.707

Table 1 shows why recall was critical for corridor construction. RF baseline was precise but missed many path nodes, while balanced RF improved recall at the cost of selectivity. GraphSAGE achieved the strongest recall and F1-score, making it the most suitable predictor for corridor construction.

However, classifier scores alone do not determine path-planning performance, so the final comparison used corridor-level outcomes such as certification, visited nodes, win/tie/loss rates, corridor ratio, and path recall.

Table 2. Search-level comparison of the main corridor-guided configurations.

Model	Certified	ML better	Tie	A* better	Failed	Avg. ML visited	Avg. A* visited	Corridor ratio	Path recall
Oracle corridor	1.00	0.55	0.45	0.00	0.00	4.47	6.75	0.14	1.00
GraphSAGE compact (τ=0.05)	0.99	0.07	0.92	0.00	0.01	6.57	6.75	0.62	1.00
GraphSAGE compact (τ=0.30)	0.94	0.24	0.70	0.00	0.07	5.60	6.75	0.29	0.98
GraphSAGE robust (τ=0.30)	1.00	0.24	0.70	0.06	0.00	8.36	6.75	0.29	0.98
RF baseline (τ=0.08)	1.00	0.06	0.81	0.13	0.00	14.59	6.75	0.30	0.96

Table 2 shows that the oracle corridor provides the strongest upper bound, improving over global A in 55% of cases and tying in the remaining cases. Among the learned models, GraphSAGE compact at threshold 0.30 gave the strongest efficiency result, reducing average visited nodes from 6.75 to 5.60 and improving over A* in 24% of cases. However, it also produced 7% uncertified cases, showing that compact corridors can still miss important path nodes.*

GraphSAGE robust removed uncertified cases by using stronger adaptive recovery settings (`max_expansions=11`, `release_k=5`). This achieved 100% certification, but increased average visited nodes to 8.36 because more blocked candidates could be released during recovery. This confirms the main trade-off of the approach: stronger expansion improves reliability, but can reduce the efficiency benefit.

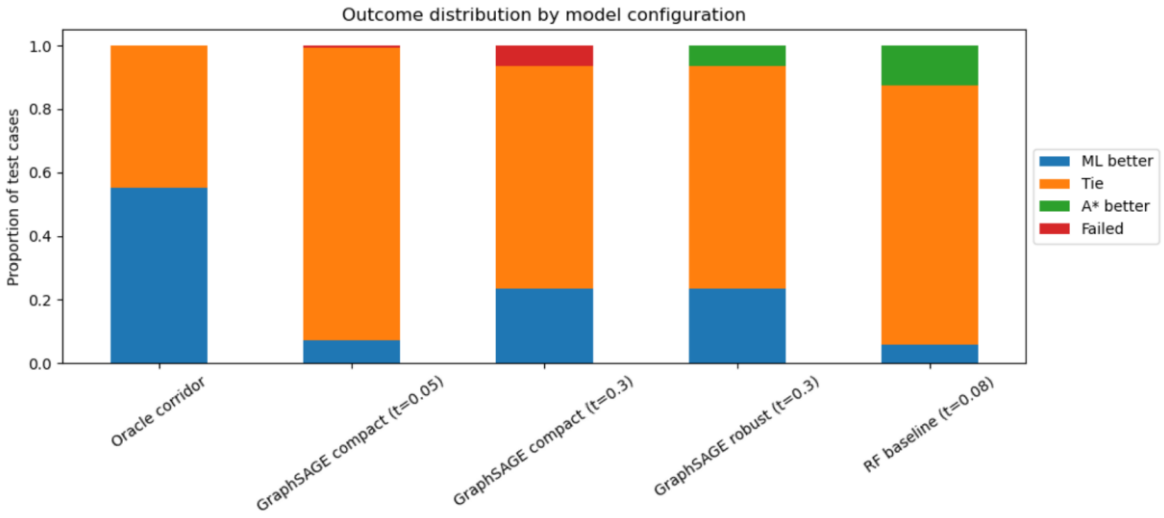

Figure 4. Outcome distribution across corridor-guided configurations.

Figure 4 shows that ties form a large proportion of outcomes. A tie is not a failure: it means the corridor-guided method certified the same optimal path while visiting the same number of nodes as global A. GraphSAGE compact at threshold 0.30 gives the strongest learned-model efficiency result. GraphSAGE robust removes failed cases, but introduces more A* better outcomes because stronger recovery increases search effort. RF baseline produces more overhead cases, confirming that its corridor is less efficient for search.*

5.3 Case-Level Analysis

The aggregate results show the overall performance trend, but individual graph cases help explain why the same architecture can produce wins, ties, or losses. The following examples illustrate the three main certified outcomes observed during testing.

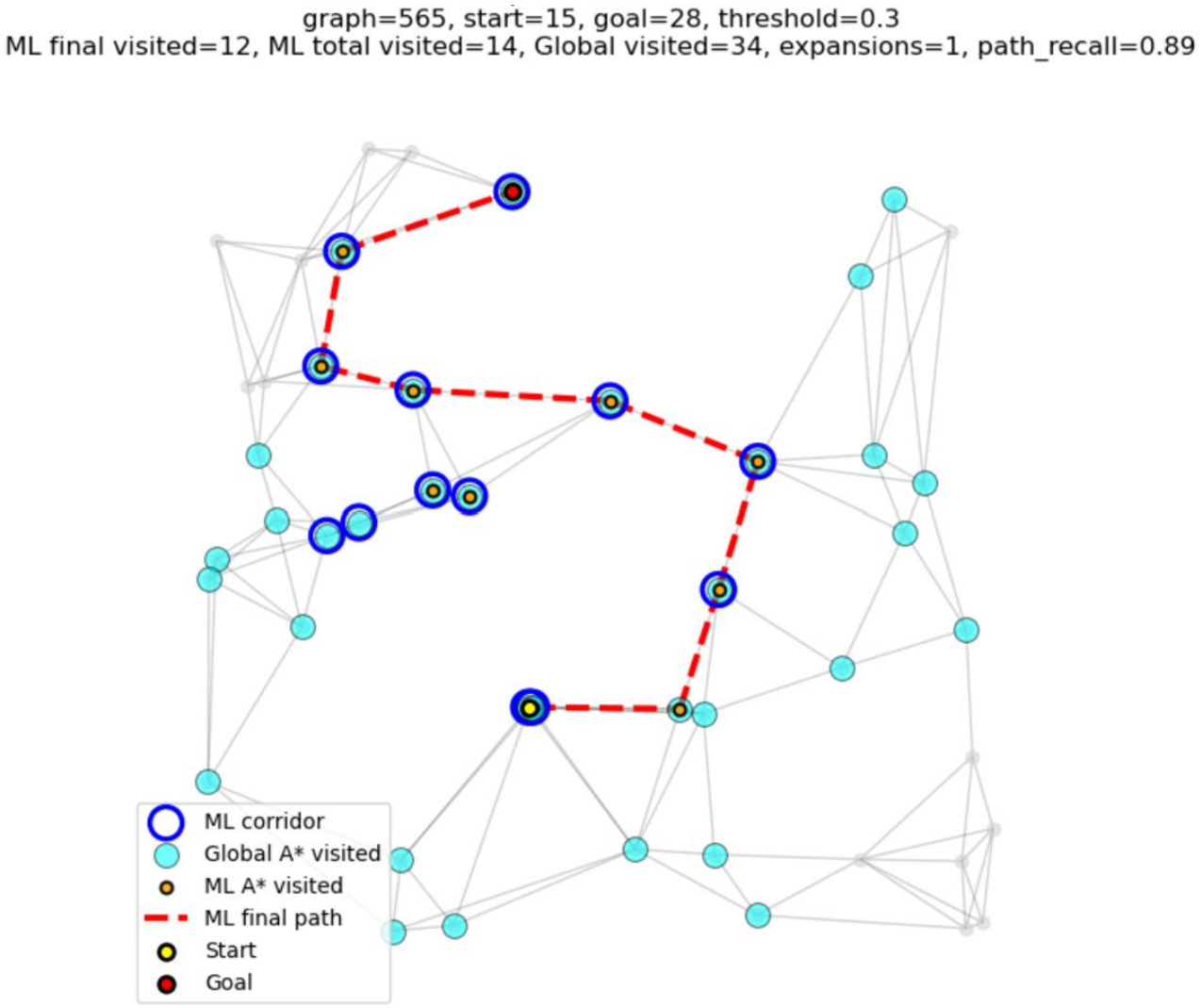

Figure 5. GraphSAGE win case.

Corridor-guided A* finds the certified path with 14 total visited nodes, compared with 34 for global A*. One adaptive expansion is required, showing that recovery can improve completeness while still preserving a search-space reduction.

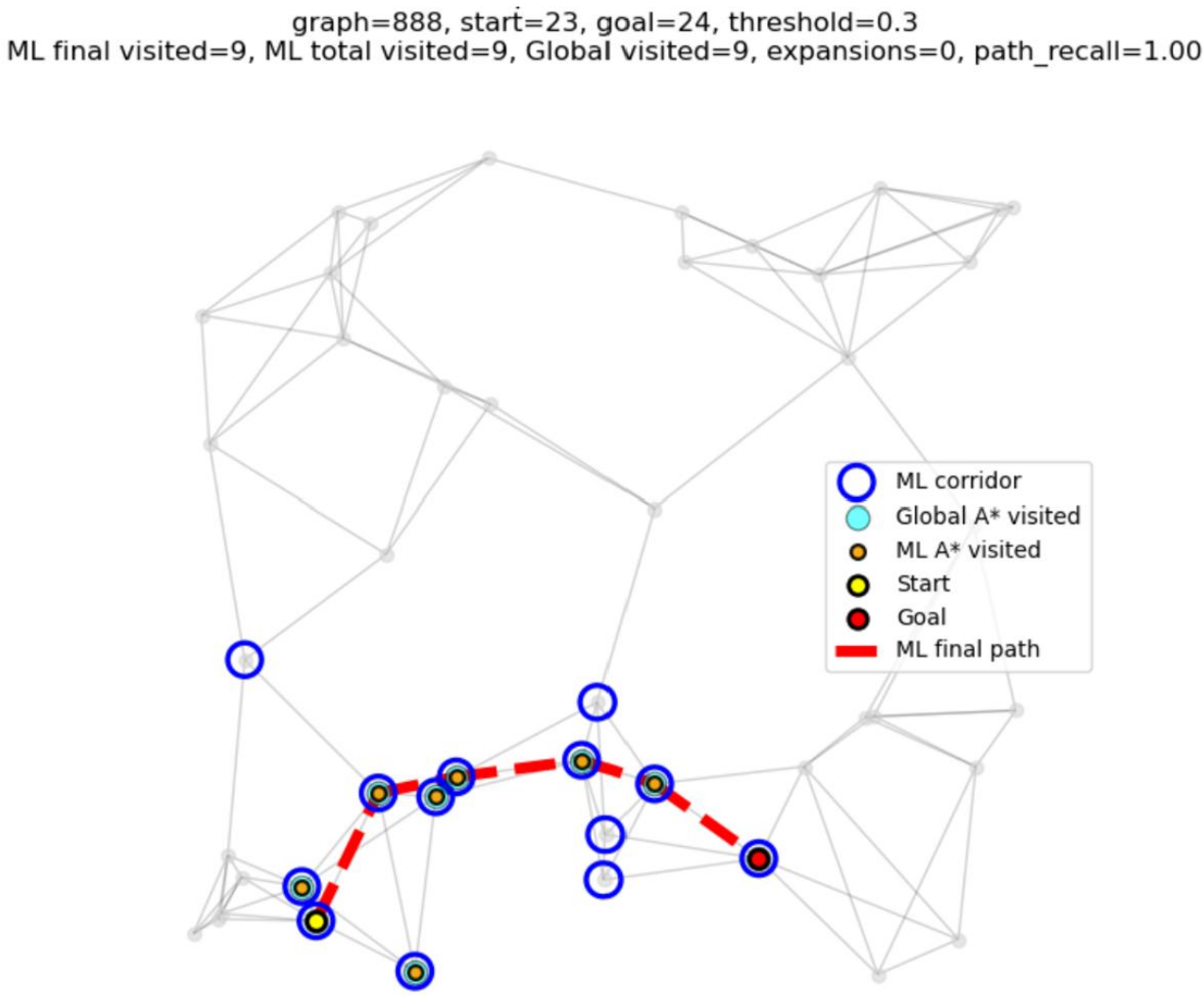

Figure 6. GraphSAGE tie case.

Corridor-guided A* and global A* both visit 9 nodes, with no adaptive expansion required. The path is certified, but the corridor does not reduce search effort in this case.

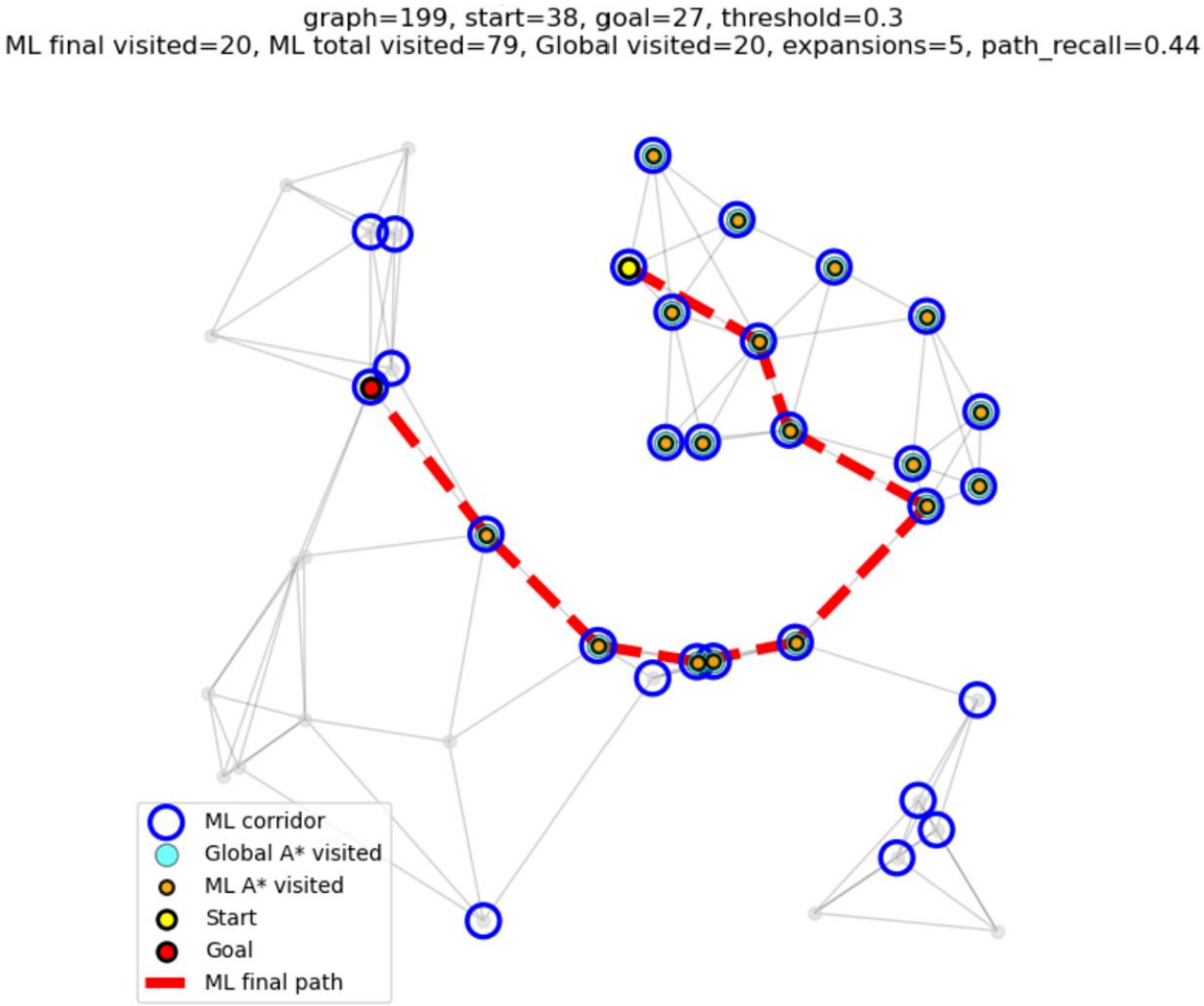

Figure 7. GraphSAGE overhead case.

Although the final corridor-guided search pass visits the same number of nodes as global A*, the total ML search visits 79 nodes because five adaptive expansion steps are required. This illustrates how robust recovery can preserve certification while increasing search effort.

6. Limitations and Future Improvements

Corridor prediction could be improved using more expressive learning models that better capture graph structure and long-range dependencies. This could help produce corridors that are both compact and complete, reducing the need for adaptive expansion.

Corridor threshold could be selected adaptively rather than fixed manually. The results showed a clear trade-off between compact corridors and path recall. A low threshold increased coverage but produced many ties, while a higher threshold reduced the corridor size but increased the risk of uncertified cases. A future system could adjust the threshold based on graph size, start-goal distance, prediction confidence, or expected corridor ratio.

Adaptive expansion could be made more selective. In the current prototype, blocked candidates are released according to search priority, but future versions could combine A* priority with model probability, local connectivity, and distance-to-goal information. This could reduce unnecessary releases while still recovering missing path nodes.

The approach should also be tested on larger graphs and more diverse navigation scenarios. Integration with a robot navigation stack could further evaluate its practical applicability for autonomous mobile robot navigation.

7. Conclusion and Future Trends

The results show that the approach is feasible, but strongly dependent on corridor quality. GraphSAGE provided the strongest learned-model result, while the oracle corridor showed that larger reductions are theoretically possible when the corridor is accurate.

The main limitation is the trade-off between compactness and completeness. Compact corridors can reduce visited nodes, but may miss required path nodes and trigger adaptive expansion. Robust recovery improves certification, but can increase search effort. This shows that corridor-guided search is most effective when the prediction model achieves high path recall without making the corridor too large.

Future work should move toward fixed, persistent navigation maps rather than independently generated graphs. In such settings, the model could learn recurring route patterns from repeated queries over the same environment. Subgraph segmentation or hierarchical corridor prediction could also make the method more scalable and closer to real mobile robot navigation, supporting faster planning decisions while preserving the reliability of classical graph search.

References

Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) 'A formal basis for the heuristic determination of minimum cost paths', IEEE Transactions on Systems Science and Cybernetics, 4(2), pp. 100–107. https://doi.org/10.1109/TSSC.1968.300136

Che N, Zeng X, Zhao J, Wang H, Du Q. An Enhanced Hybrid Astar Path Planning Algorithm Using Guided Search and Corridor Constraints. *Sensors*. 2026; 26(2):379. https://doi.org/10.3390/s26020379

Hamilton, W.L., Ying, R. and Leskovec, J. (2017) 'Inductive representation learning on large graphs', Advances in Neural Information Processing Systems, 30. https://dl.acm.org/doi/10.5555/3294771.3294869

Pándy, M., Qiu, W., Corso, G., Veličković, P., Ying, Z., Leskovec, J. and Liò, P. (2022) 'Learning Graph Search Heuristics', Proceedings of the First Learning on Graphs Conference, Proceedings of Machine Learning Research, 198, pp. 10:1–10:13. Available at: https://proceedings.mlr.press/v198/pandy22a.html (Accessed: 13 June 2026).

Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M. and Kanezaki, A. (2021) 'Path Planning using Neural A* Search', Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research, 139, pp. 12029–12039. Available at: https://proceedings.mlr.press/v139/yonetani21a.html (Accessed: 13 June 2026).

Mobile Industrial Robots (2026) MiR Fleet. Available at: https://mobile-industrial-robots.com/products/software/mir-fleet (Accessed: 13 June 2026).

Amazon (2024) 'I'm Amazon's first autonomous robot. Follow me around on my workday'. Available at: https://www.aboutamazon.com/stories/amazon-robotics-autonomous-robot-proteus-warehouse-packages (Accessed: 13 June 2026).

Open Navigation LLC (2026) Nav2 documentation. Available at: https://docs.nav2.org/ (Accessed: 13 June 2026).

Macenski, S., Martín, F., White, R. and Ginés Clavero, J. (2020) 'The Marathon 2: A Navigation System', arXiv:2003.00368. Available at: https://arxiv.org/abs/2003.00368 (Accessed: 13 June 2026).