International Journal on Engineering Technology and Infrastructure Development (InJET-InDev)
www.lec.edu.np/journal
Volume 2, Issue no. 2, Page 150-161
Received Date: 11/30/2025
Revised Date: 5/11/2026
Accepted Date: 5/17/2026
ISSN: 3021-999X (Print)
https://doi.org/10.3126/injet-indev.v2i2.95721

Hard-Exclusion Bidirectional A*: Constraint-Safe, Real-Time Pathfinding for Dynamic Urban Networks

Anish Dangol[1], Sudip Adhikari[2], Pralhad Chapagain[3*]

[1]*D.A.V. College, Lalitpur, Nepal, dangol.anish001@gmail.com*
[2]*D.A.V. College, Lalitpur, Nepal, sudip.adhikari48@gmail.com*
[3]*Kantipur Engineering College, Lalitpur, Nepal, pralhadchapagain@kec.edu.np*

Abstract

Urban navigation systems in dynamic environments face a fundamental challenge: delivering real-time, verifiably safe, and efficient pathfinding solutions that adapt to unpredictable obstacles while maintaining computational feasibility. Existing approaches either rely on static precomputation unsuitable for dynamic updates (Contraction Hierarchies) or employ soft-penalty methods incapable of guaranteeing strict obstacle avoidance. This paper introduces an enhanced Bidirectional A* algorithm incorporating a hard exclusion mechanism that ensures provable safety by completely pruning paths violating user-defined safety margins around dynamic obstacles. Integrated with k-d tree spatial indexing for efficient proximity queries, the algorithm achieves average execution times ranging from 0.80 ms to 4,189 ms depending on obstacle density and avoidance radius, with the majority of constrained configurations completing under 2 seconds, while low-density configurations may require up to 4 seconds due to extensive search space exploration for real-time deployment. The system is implemented in Python 3.9, leveraging OSMnx for graph construction and scipy.spatial.cKDTree for spatial indexing. Empirical evaluation on the complex urban networks of Kathmandu and Lalitpur across 1,600 experimental configurations systematically varies obstacle density and avoidance radius to quantify critical trade-offs among safety, path efficiency, and reliability. Results demonstrate that the algorithm maintains high reliability (≤20% failure rate) under moderate constraints, exhibits "fail-fast" behavior in overly constrained scenarios, and exhibits Path Length Inflation of 2.685× at most for safety-critical applications. The Path Vulnerability Index further reveals that network topology fundamentally influences feasible routing under dynamic disruption. These findings provide a validated framework for safety-critical navigation systems, enabling configurability across operational requirements while maintaining verifiable guarantees that distinguish this approach from existing penalty-based methods and establishing a foundation for next-generation intelligent transportation systems.

Keywords: Real-time pathfinding, Safety-critical navigation, Hard exclusion mechanism, k-d tree indexing, Verifiable safety guarantees

1. Introduction

Modern urban centers increasingly struggle with severe congestion that harms economic productivity, environmental quality, and overall quality of life. These challenges are especially acute in rapidly growing cities like Kathmandu and Lalitpur, where rising populations and vehicle ownership overwhelm existing road networks. The unpredictability of accidents, sudden closures, and temporary hazards makes navigation systems reliant on static or historical data inadequate (Wan, Ghazzai and Massoud, 2019) (Yan, Wenbin and Hu, 2019). Consequently, there is a critical need for real-time, safety-aware navigation solutions for such dynamic urban environments (Isa, Mohamed and Yusoff, 2015).

This research addresses the need for a pathfinding algorithm capable of producing real-time, verifiably safe, and efficient routes in dense urban networks with dynamic obstacles. High-performance methods like Contraction Hierarchies fail in such environments due to their reliance on static precomputation, while adaptable heuristic approaches like A* often incur high computational costs and lack mechanisms for enforcing strict safety margins (Foeada, Ghifaria and Kusu, 2021) (Zhang et al., 2025). This reveals a critical research gap: the absence of a solution that simultaneously ensures computational speed, path efficiency, search reliability, and provable obstacle avoidance within defined safety boundaries (Song et al., 2018; Foeada, Ghifaria and Kusu, 2021).

To bridge this gap, this paper introduces an enhanced Bidirectional A* algorithm engineered for real-time urban navigation. Its core innovation is a hard exclusion mechanism that is distinct from soft-penalty methods and

*Corresponding Authors

removes any path violating a user-defined safety margin around dynamic obstacles, ensuring verifiable safety (Yan, Wenbin and Hu, 2019).Combined with efficient spatial indexing for rapid proximity queries, the algorithm is designed to balance real-time performance, path efficiency, search reliability, and provable safety. The paper presents the development and validation of this exclusion strategy, a quantitative analysis of its multi-objective trade-offs under varying obstacle densities, and an empirical demonstration of its feasibility on a complex urban network, providing a robust framework for next-generation navigation systems.

2. Related Works

Pathfinding in dynamic urban environments is a complex problem at the intersection of classical graph search, real-time systems, and dynamic obstacle avoidance. Navigation systems in such settings must respond quickly to changing conditions while ensuring safety and efficiency. This section reviews prior work organized into four thematic areas: classical pathfinding algorithms, real-time adaptations and dynamic obstacle handling, hybrid and heuristic-based methods, and practical implementations with data integration. This organization highlights the contributions and limitations of existing approaches and clarifies how the current work fits within this context.

The A* algorithm remains foundational for pathfinding due to its completeness and optimality under admissible heuristics. However, classical A* suffers from performance and memory limitations in large or complex environments (Foeada, Ghifaria and Kusu, 2021). Variants such as Multi-Heuristic A* (MHA*) address challenges like heuristic depression regions by running multiple simultaneous searches guided by both admissible and inadmissible heuristics, achieving faster convergence than Weighted A* in complex robotics domains (Aine et al., 2016). Dynamic weighting methods have been proposed to adjust the relative emphasis of heuristic and cost functions as the search progresses, prioritizing heuristic guidance early and actual cost near the goal, effectively reducing the number of traversed nodes and overall search time (Li et al., 2023). Additional optimizations, such as marking blocked nodes as impassable instead of backtracking, have demonstrated up to a 70% reduction in path-planning time (Liu, Wang and Xu, 2022). While these enhancements improve computational efficiency and scalability, they do not provide explicit mechanisms to guarantee strict avoidance of dynamic obstacles, a key limitation addressed by the hard exclusion mechanism introduced in this research.

Dynamic obstacle avoidance is a central challenge in real-time urban navigation. Classical methods often model obstacles as sources of cost or risk, with the Artificial Potential Field (APF) method guiding agents via attractive forces from the goal and repulsive forces from obstacles. Although APF is effective in simple scenarios, it is prone to local minima, limiting its applicability in dense, complex environments. Hybrid approaches combine a global planner such as A* with local reactive methods like APF or the Dynamic Window Approach (DWA) (Zhang et al., 2025) (Liu, Wang and Xu, 2022). In these architectures, the global planner provides strategic guidance by generating a feasible path, while the local planner reacts to unforeseen obstacles in real time, ensuring local safety. Similarly, vehicular ad-hoc network studies often model traffic density and signal delays as dynamic cost penalties to steer routing decisions toward safer and faster routes (Jerbi et al., 2006) (Yan, Wenbin and Hu, 2019). Real-time crowd-sourced reports of blocked roads are also incorporated, adding large penalties to obstructed paths to discourage selection (Wan, Ghazzai and Massoud, 2019). These methods improve adaptability and responsiveness, but they rely on soft penalties that cannot provide verifiable guarantees that paths fully avoid obstacles.

Heuristic and hybrid methods extend A* to improve robustness and adaptability in dynamic environments. Self-adaptive evolutionary algorithms for Dynamic Vehicle Routing Problems (DVRP) incorporate dynamic traffic factors into travel times, producing solutions that remain robust under changing conditions (Sabar et al., 2019). Multi-agent frameworks allow individual vehicles to learn from experience and select routes optimizing utility functions that account for driver preference, cost, and uncertainty due to unexpected incidents (Yan, Wenbin and Hu, 2019). While these approaches demonstrate strong adaptability and context awareness, they often involve trade-offs between computational efficiency and strict enforcement of safety margins, limiting their applicability for real-time navigation with provable guarantees.

Urban networks present unique challenges due to recurrent and non-recurrent congestion patterns (Isa, Mohamed and Yusoff, 2015). Existing solutions integrate traffic models, crowd-sourced data, and multi-layer planning architectures to improve real-time responsiveness (Wan, Ghazzai and Massoud, 2019). Although these systems can adapt to changing conditions, a key limitation remains: most rely on soft penalties or reactive local planners

that cannot guarantee that paths fully avoid dynamic obstacles. This research addresses this gap by integrating a hard exclusion mechanism into a Bidirectional A* framework, ensuring verifiable safety while maintaining real-time performance, path efficiency, and reliability in complex urban networks.

3. Methodology

3.1. Rationale for Core Design Choices

The primary contribution of this work is an enhanced Bidirectional A* algorithm incorporating a hard exclusion strategy to guarantee verifiable safety in real-time urban navigation with dynamic obstacles. To achieve this, the system integrates bidirectional search for computational efficiency and explicitly avoids Contraction Hierarchies due to their inflexibility in dynamic environments.

Bidirectional A* was chosen over Dijkstra and unidirectional A* for its efficiency in large, complex networks. While Dijkstra guarantees shortest paths, its uniform exploration is costly, and standard A* improves efficiency using a heuristic. Bidirectional search further accelerates computation by exploring from both source and destination, typically meeting near the midpoint. Prior studies show heuristic-guided and bidirectional methods reduce node expansions and execution time in complex domains (Foeada, Ghifaria and Kusu, 2021) (Aine et al., 2016) (Liu, Wang and Xu, 2022), making them suitable for real-time pathfinding.

Contraction Hierarchies (CH), though offering fast query times in static networks, rely on extensive precomputation and are therefore ill-suited for dynamic environments where edge or node traversability can change frequently. Bidirectional A* provides the flexibility to incorporate real-time obstacle updates without recomputing the entire graph. The hard exclusion strategy forms the core innovation of this approach. Unlike soft-penalty methods that merely increase edge costs, hard exclusion completely prunes any path violating a user-defined safety margin. This ensures that all returned paths comply with the specified avoidance radius, providing a provable safety guarantee essential for safety-critical applications.

3.2. System Architecture Overview

The pathfinding system is organized as a modular, service-oriented architecture to support real-time navigation in dynamic urban networks. Its primary components include the Service Interface, Pathfinding Core, Spatial Index, and Dynamic Obstacle Cache, all interacting with the preprocessed graph data G. A query workflow begins when a user specifies a source, destination, and safety parameter (σ). The Service Interface validates the inputs and translates σ into a concrete obstacle radius. The Pathfinding Core then executes the enhanced Bidirectional A* algorithm, consulting the Spatial Index (k-d tree) for efficient proximity checks and the preprocessed graph G for neighbor and edge attributes. Active obstacles are maintained in the Dynamic Obstacle Cache, which is periodically updated from the Persistent Obstacle Store to reflect real-time changes in the environment. The Pathfinding Core returns either a safe path or a failure signal to the Service Interface, which delivers results back to the query source. This architecture separates preprocessing, caching, and per-query operations, enabling modularity, maintainability, and real-time performance.

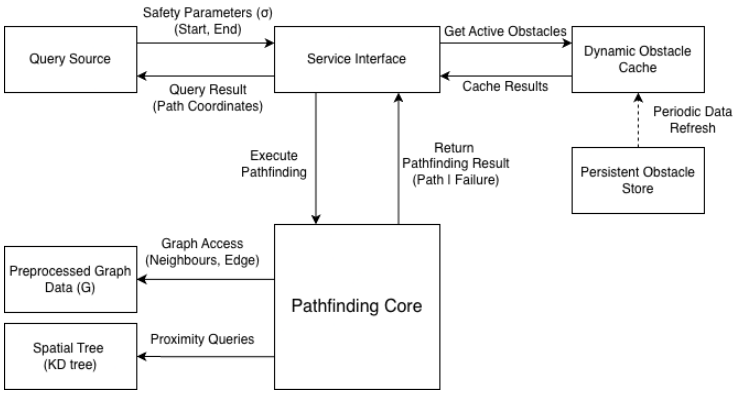

Figure 1. Architectural Design of the System

3.3. Data Acquisition and Graph Representation

The urban road networks of Kathmandu and Lalitpur, Nepal, were used as a complex testbed for real-time pathfinding under dynamic conditions. Road data from OpenStreetMap (OSM) was processed with OSMnx to construct a directed, weighted graph $G=(V,E)$, where vertices V represent intersections and dead-ends with geographic coordinates, and edges E represent road segments weighted by length and annotated with road type and geometry to capture curves accurately. This high-fidelity representation allows the algorithm to consider spatial and topological features, applying safety margins along curved segments and supporting systematic evaluation of efficiency (path length, execution time) and safety (proximity to dynamic obstacles) across varying obstacle densities and avoidance radii. By incorporating both geometric and topological fidelity, the graph provides a robust foundation for assessing algorithmic trade-offs in safety-critical urban navigation.

3.4. Spatial Indexing for Efficient Proximity Queries

Real-time pathfinding in dense urban networks requires rapid evaluation of node and edge proximity to dynamic obstacles. A naïve linear scan of all nodes would incur O(|V|) complexity per query, which is prohibitive for real-time performance. To address this, a k-d tree (Bentley, 1975) was constructed using the latitude and longitude of each node, enabling O(log |V|) average-case nearest-neighbor queries. This allows arbitrary coordinates to be efficiently mapped to the nearest graph nodes, supporting fast distance calculations to nearby obstacles. Edges are indexed using a hash map for constant-time access to attributes such as length, geometry, and connectivity, ensuring efficient adjacency queries required by the hard exclusion mechanism.

The node index is built at system initialization, while dynamic obstacles are managed separately in a Dynamic Obstacle Cache, allowing updates without reconstructing the k-d tree. This separation preserves computational efficiency during search expansions. Similar indexing strategies in prior dynamic pathfinding research have demonstrated significant reductions in query overhead while maintaining accurate obstacle avoidance (Liu, Wang and Xu, 2022). From a research perspective, this framework provides a scalable, reproducible foundation for systematic evaluation, enabling experiments across varying obstacle densities and avoidance radii without confounding algorithmic performance with query computation time.

3.5. Enhanced Bidirectional A Algorithm with Hard Exclusion*

The central contribution of this work is an enhanced Bidirectional A* algorithm designed to generate real-time paths in dense urban networks while strictly enforcing user-defined safety constraints. The algorithm extends the classical Bidirectional A* search by integrating a hard exclusion mechanism, ensuring that no path violates a predefined safety margin around dynamic obstacles. This approach provides provable safety guarantees, which cannot be reliably achieved by soft-penalty methods that merely increase edge costs.

Baseline Bidirectional A*

The algorithm performs simultaneous searches from both the source and destination nodes. A meeting node is identified when the two search frontiers converge at a common node, after which the complete path is reconstructed by tracing the forward and backward parent pointers (Kaindl and Kainz, 1997) (Goldberg and Harrelson, 2005). This bidirectional exploration reduces the total search space and execution time relative to unidirectional A*, enabling real-time performance in large, complex urban networks.

Hard Exclusion Mechanism

The hard exclusion strategy is the core innovation of the algorithm. Dynamic obstacles are represented as a set of impassable nodes $O \subset V$, and each query includes a user-defined obstacle_radius, specifying a circular safety margin around each obstacle. Four discrete radius presets were evaluated in this study:

Table 1. Obstacle Radius Presets and Corresponding Safety Margins

Obstacle Setting	Radius (km)	Radius (m)
Tight	0.05	50
Standard	0.10	100
Wide	0.20	200
Very Wide	0.50	500

These presets allow systematic evaluation of the safety-efficiency trade-off across a range of real-world avoidance scenarios, from minimal clearance around small hazards to broad exclusion zones around major incidents. Candidate neighbor nodes and edges are evaluated through a three-stage validation process:

Node Obstacle Check (O(1)): Neighbor nodes belonging to O are immediately discarded.

Node Proximity Check (O(log|O|)): A pre-built k-d tree over obstacle coordinates is queried to determine whether any obstacle lies within the obstacle_radius of a candidate neighbor node. Nodes within this radius are excluded to prevent unsafe terminations.

Edge Proximity Check (O(log|O|)): Each edge is represented as a LineString. The midpoint and both endpoints of the edge are queried against the obstacle k-d tree. Edges for which any of these points fall within the obstacle_radius of any obstacle are discarded, ensuring no path segment violates the safety margin.

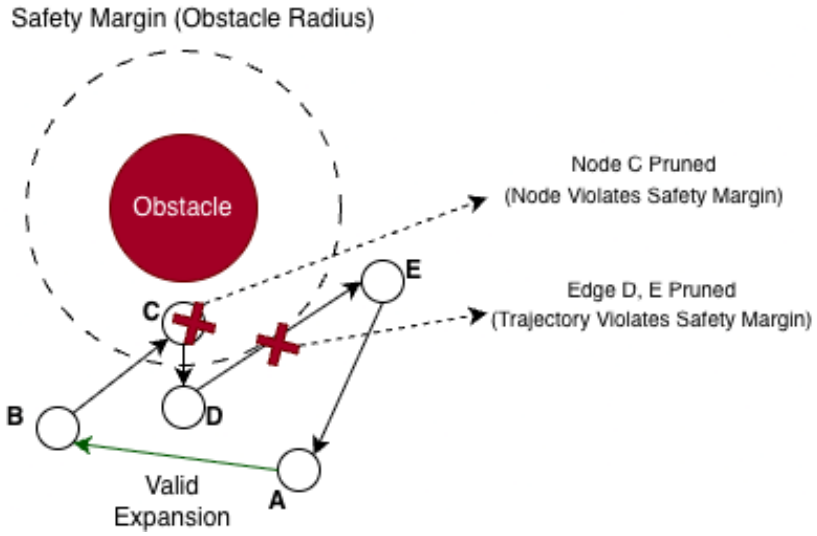

Figure 2. Hard Exclusion Mechanism

Both the Node Proximity Check and Edge Proximity Check are accelerated by a secondary k-d tree constructed over active obstacle coordinates at query initialization. This reduces per-expansion obstacle query complexity from O(|O|) to O(log|O|), ensuring consistent sub-second average execution times even at the highest tested obstacle density of ρ=500.

Path Vulnerability Index (PVI)

To quantify residual exposure of a returned path to obstacle proximity, the Path Vulnerability Index is defined as the proportion of total path length that passes within a vulnerability buffer of 1.5× the obstacle_radius around any active obstacle:

$$PVI = (L_{vulnerable} / L_{total}) \times 100 \quad \text{(Equation 1)}$$

where L_total is the total length of the path in meters, and L_vulnerable is the cumulative length of path edges for which any point falls within 1.5 × obstacle_radius of any obstacle. A PVI of 0 indicates complete separation from all obstacles, while higher values indicate greater residual proximity. Failed paths (where no valid route exists) are assigned a sentinel value of −1 and excluded from aggregate PVI calculations.

Path Reconstruction and Failure Handling

When the forward and backward frontiers meet at a meeting node, the path is reconstructed by tracing back the came_from pointers to the source and destination. If both search frontiers are exhausted without meeting, the algorithm returns None, indicating that no valid path exists under the current safety constraints. This mechanism allows for systematic evaluation of reliability under varying obstacle densities and safety margins.

3.6. Notation and Definitions

The following metrics and parameters are used throughout this study:

Parameters:

- **ρ (obstacle density):** The number of obstacle nodes randomly placed in the graph, taking values of 50, 150, 300, and 500.
- **r (obstacle radius):** The spatial avoidance margin around each obstacle in kilometers, corresponding to the presets defined in Table 1.
- **σ (safety parameter):** The user-defined safety level input that the Service Interface translates into a concrete obstacle radius r.

Metrics:

- **Failure Rate (FR):** The proportion of test runs in which no valid path was found under the given constraints, expressed as a percentage.
- **Path Length Inflation (PLI):** The ratio of the length of the safe path found by the enhanced algorithm to the length of the baseline path found without obstacle avoidance. A PLI of 1.0 indicates no detour; higher values indicate longer detours.
- **Path Vulnerability Index (PVI):** As defined in Section 3.5, the proportion of total path length passing within 1.5× the obstacle radius of any active obstacle, expressed as a percentage.
- **Search Space Reduction (SSR):** The percentage reduction in the number of nodes expanded by the enhanced algorithm relative to the baseline, indicating computational savings from early pruning.

3.7 Implementation Details

The algorithm and all experimental components were implemented in Python 3.9. The road network graph was constructed and preprocessed using OSMnx, which interfaces directly with the OpenStreetMap API to retrieve and process geographic data. Graph operations and neighbor queries were performed using NetworkX. Spatial indexing and k-d tree construction were implemented using scipy.spatial.cKDTree, and geometric edge proximity checks were performed using the Shapely library's LineString and Point primitives. All experiments were executed on a standard consumer laptop, and results were recorded to CSV for reproducibility. The experimental script, graph data, and configuration files are available upon request from the corresponding author.

4. Results

This section presents the empirical evaluation of the enhanced Bidirectional A* algorithm. A total of 1,600 runs were performed across 16 experimental configurations, systematically varying obstacle density (ρ) and avoidance radius (r). For each configuration, 100 source-destination pairs were generated by random sampling of graph nodes using a fixed random seed (seed=42), ensuring reproducibility across all runs while maintaining diversity in path characteristics. Performance is benchmarked against a baseline Bidirectional A* algorithm without obstacle avoidance to quantify the impact of the hard exclusion mechanism. The analysis focuses on multiple metrics: Failure Rate (FR) for reliability, Path Length Inflation (PLI) for efficiency, Path Vulnerability Index (PVI) for safety, and Execution Time (ET) for computational cost. This setup enables a rigorous assessment of the trade-offs between safety, path optimality, and real-time performance under dynamic urban conditions.

4.1 Summary of Aggregate Results

A high-level overview of the algorithm's performance across all 16 experimental configurations is presented in Table 2. This data is aggregated directly from the 1,600 individual test runs. The table clearly shows that as both obstacle density (ρ) and avoidance radius (r) increase, there is a corresponding increase in failure rate and path length inflation. Conversely, the Path Vulnerability Index (PVI) and Search Space Reduction (SSR) also show distinct patterns, which are explored in the subsequent sections. The baseline Bidirectional A* exhibited an average execution time of approximately 273 ms across test configurations, as it performs no obstacle checks. The enhanced algorithm outperforms this baseline in highly constrained configurations where fast failure dominates, while incurring higher execution times in low-density configurations due to the overhead of proximity checks.

Table 2. Aggregate Result

Obstacle Density (ρ)	Radius Setting	Avg. PVI (%) ± SD	Avg. Time (ms) ± SD	Avg. SSR (%) ± SD	Avg. Path Inflation ± SD	Baseline Time (ms)	Failure Rate (%)
50	tight	0.70 ± 1.71	2607.96 ± 2097.29	59.96 ± 4.62	1.027 ± 0.080	273.49	1.0
	standard	2.56 ± 4.84	4189.43 ± 3815.64	60.86 ± 4.95	1.034 ± 0.088	273.49	2.0
	wide	6.65 ± 7.88	4178.76 ± 3939.25	62.94 ± 17.97	1.082 ± 0.163	273.49	9.0
	very wide	27.71 ± 21.16	2177.03 ± 2731.93	76.91 ± 32.69	1.234 ± 0.320	273.49	42.0
150	tight	1.31 ± 2.59	3091.98 ± 3169.50	59.61 ± 31.33	1.028 ± 0.107	273.49	7.0
	standard	6.99 ± 8.83	3020.60 ± 3188.87	66.07 ± 12.32	1.056 ± 0.128	273.49	11.0
	wide	16.32 ± 14.24	1477.27 ± 2187.01	73.62 ± 21.79	1.086 ± 0.143	273.49	39.0
	very wide	38.74 ± 28.36	58.42 ± 144.25	95.19 ± 14.61	1.912 ± 1.076	273.49	94.0
300	tight	2.58 ± 3.56	732.31 ± 637.36	65.29 ± 11.54	1.054 ± 0.103	273.49	12.0
	standard	7.84 ± 7.87	475.33 ± 499.47	71.56 ± 17.64	1.079 ± 0.119	273.49	26.0
	wide	26.00 ± 17.75	268.42 ± 383.36	68.56 ± 78.32	1.227 ± 0.263	273.49	67.0
	very wide	9.25 ± 0.00	2.75 ± 15.52	99.73 ± 2.15	2.685 ± 0.000	273.49	99.0
500	tight	6.49 ± 5.81	459.63 ± 407.58	62.12 ± 34.59	1.128 ± 0.175	273.49	15.0
	standard	17.98 ± 10.17	335.64 ± 379.80	54.10 ± 132.27	1.271 ± 0.375	273.49	44.0
	wide	46.07 ± 22.47	57.26 ± 122.08	79.69 ± 155.43	1.447 ± 0.445	273.49	94.0
	very wide	N/A	0.80 ± 3.83	99.87 ± 1.29	N/A	273.49	100.0

Note: PVI and Path Length Inflation are reported only for successful pathfinding runs. Configurations with 100% failure rates are reported as N/A because no valid paths exist for evaluation. At very high failure rates, the small subset of successful paths tends to remain farther from obstacles, resulting in comparatively low PVI values among successful runs. At very high failure rates, the small subset of successful paths tends to remain farther from obstacles, resulting in comparatively low PVI values among successful runs.

4.2. Reliability Analysis: Failure Rate (FR)

The Failure Rate (FR) is a critical metric for understanding the limits of the hard exclusion approach. Figure 3 illustrates the percentage of queries that failed to find a valid path for each experimental configuration.

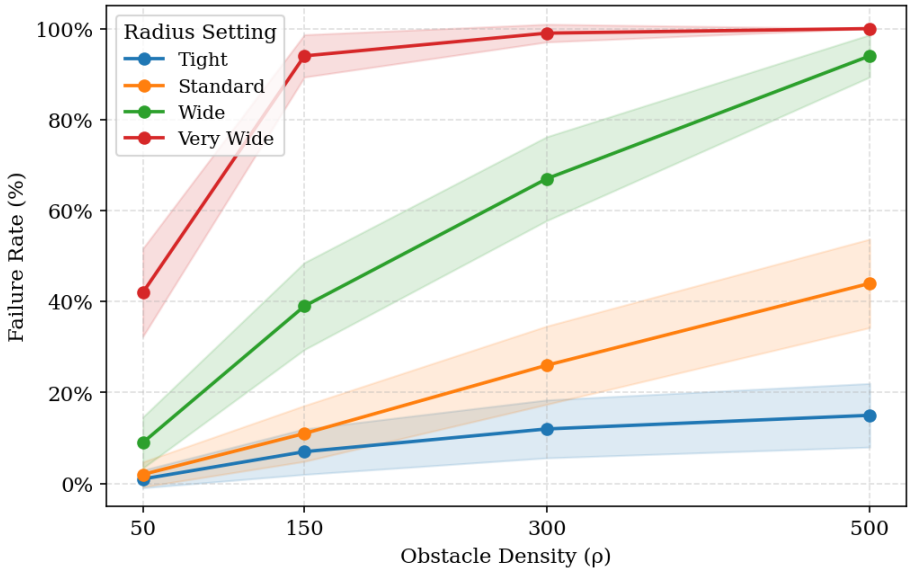

Figure 3. Algorithm failure rate as a function of obstacle density, categorized by avoidance radius setting. Shaded regions represent 95% confidence intervals.

As shown in the figure, the FR increases with both obstacle density and the avoidance radius. With the 'tight' radius, the algorithm demonstrates high reliability; the failure rate remains below 20% even at the highest obstacle density of 500. However, as the radius widens, the system's ability to find a path degrades significantly. The 'very_wide' radius setting proves to be highly restrictive, causing a failure rate of 42% at just 50 obstacles, which rises to 99% at 300 obstacles. This result starkly illustrates the trade-off between guaranteed safety and path availability.

4.3. Efficiency Analysis: Path Length Inflation (PLI)

Path Length Inflation measures the cost of safety in terms of increased path distance, considering only successful pathfinding attempts.

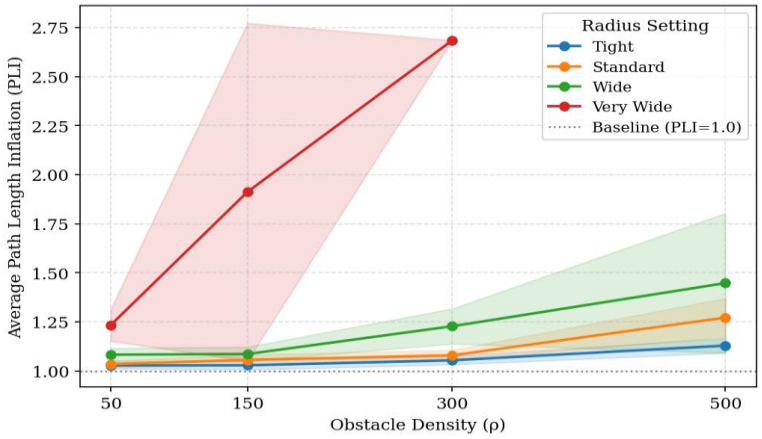

Figure 4. Average Path Length Inflation (PLI) for successful paths across different obstacle densities and radius settings. Shaded regions represent 95% confidence intervals.

Figure 4 shows that for low obstacle densities, the PLI remains minimal, especially for 'tight' and 'standard' radii, often staying close to 1.0 (no inflation). This indicates that for minor disruptions, safe alternative paths of similar length are readily available. However, as density and radius increase, the algorithm is forced to find more significant detours. For instance, at a density of 300 obstacles, the average path for the 'wide' radius setting is 22.7% longer than the baseline (PLI = 1.227), demonstrating a substantial efficiency cost for ensuring a wide safety margin in a congested environment. A notable peak occurs for the ‘very_wide’ setting at 150 obstacles (PLI

= 1.912), indicating that under these constraints, the limited set of feasible paths requires substantially longer detours relative to the baseline shortest paths.

4.4. Safety Analysis: Path Vulnerability Index (PVI)

The PVI provides a more nuanced view of path safety, measuring the proportion of a path that, while valid, lies in a "warning zone" (1.5x the avoidance radius).

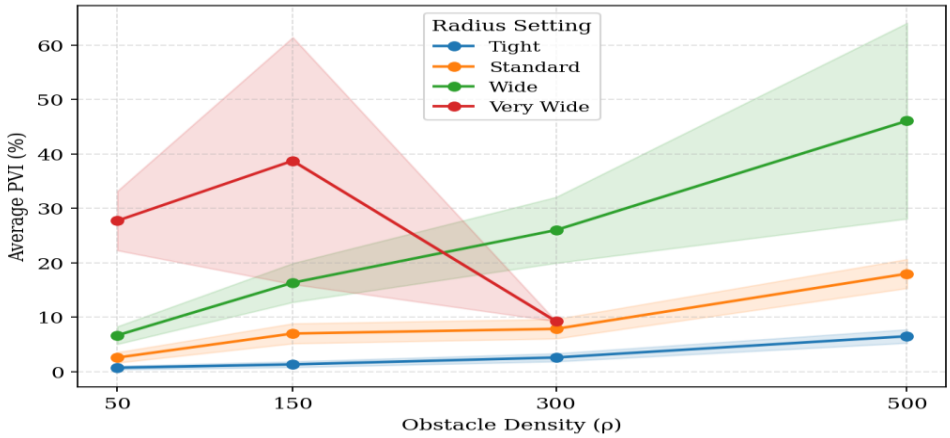

Figure 5. Average Path Vulnerability Index (PVI) for successful paths, indicating the percentage of the path within a 'warning zone'. Shaded regions represent 95% confidence intervals.

The trends in Figure 5 reveal that for the 'tight' and 'standard' settings, the average PVI is consistently low but rises with obstacle density. This suggests that as the environment becomes more cluttered, the algorithm is forced to route paths closer to obstacle warning zones. An interesting counter-trend appears for 'wide' and 'very_wide' settings at high densities. The average PVI drops significantly in these configurations. Since failed runs are excluded from PVI calculations, this reflects a genuine selection effect: at very high failure rates, the only paths successfully found are those that are inherently far from all obstacles, resulting in a lower average vulnerability among the small subset of successful routes.

4.5. Performance Analysis: Computational Cost (ET)

The execution time of the algorithm is a direct measure of its viability for real-time applications.

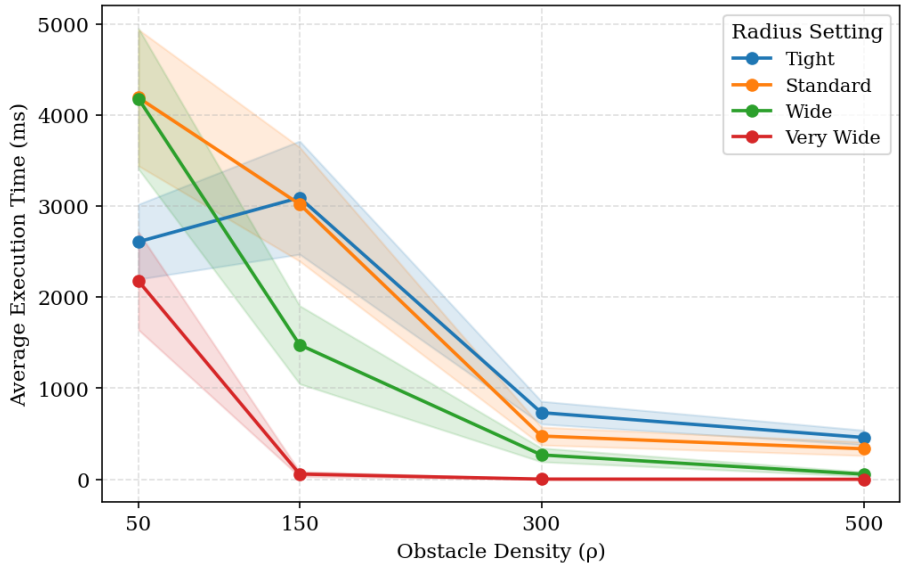

Figure 6. Average algorithm execution time versus obstacle density for each radius setting. Shaded regions represent 95% confidence intervals.

As illustrated in Figure 6 and detailed in Table 2, average execution times vary considerably across configurations, ranging from under 1 ms in highly constrained scenarios where the algorithm terminates rapidly, to over 4 seconds in low-density configurations where extensive search is required. This wide range reflects two opposing dynamics. In low-density configurations (ρ=50), the search space is largely unconstrained, requiring thorough exploration before a meeting point is found, resulting in longer execution times. Conversely, for the highest densities (ρ=300 and ρ=500) with 'wide' and 'very_wide' radii, the algorithm exhibits a dramatic decrease in average execution time. This is because the high failure rates in these configurations (as seen in Figure 3) lead to very rapid search termination — the algorithm quickly exhausts all viable paths and returns failure, bringing the average time down significantly. This highlights a key performance characteristic: the algorithm "fails fast" in overly constrained scenarios, preserving system responsiveness even when no valid path exists.

5. Discussion and Conclusion

5.1. Discussion

This study presents an enhanced Bidirectional A* algorithm with a hard exclusion mechanism for real-time pathfinding in dynamic urban environments. By strictly pruning any path that violates a user-defined safety radius, the approach provides verifiable safety guarantees while maintaining average execution times under 2 seconds for the majority of constrained configurations across 1,600 experimental runs on the Kathmandu and Lalitpur road networks. The system exhibits a "fail-fast" behavior, quickly detecting infeasible paths in highly constrained scenarios to preserve responsiveness. Empirical results quantify the trade-offs among safety, path efficiency, and reliability: increasing the avoidance radius raises path length inflation and failure rates, while smaller radii yield faster, more reliable paths. The Path Vulnerability Index further reveals that, under extreme constraints, successful paths naturally concentrate in less vulnerable regions, highlighting how network topology influences feasible routing under disruption. These insights support practical applications such as emergency vehicle routing around accident zones, disaster-response navigation during road blockages, and user-configurable safety-efficiency trade-offs in civilian navigation systems. The findings may also inform urban planning efforts aimed at identifying network bottlenecks and routing vulnerabilities under dynamic disruption.While the hard exclusion mechanism offers verifiable safety guarantees over soft-penalty approaches, this study does not include a direct empirical comparison against penalty-based methods; future work should benchmark the two strategies to quantitatively characterize the trade-off between guaranteed obstacle avoidance and path optimality.

5.2. Limitations

The current evaluation has several limitations. First, obstacles are modeled as static and uniformly distributed across the graph, which does not reflect real-world obstacle patterns such as clusters around accident sites or corridor-style closures along major roads. Results under such non-uniform distributions may differ from those reported here. Second, the evaluation is conducted on a single urban topology that are the road networks of Kathmandu and Lalitpur and generalizability to other urban morphologies (e.g., grid-based cities or larger metropolitan networks) has not been empirically verified. Third, while proximity-check complexity was reduced from O (|O|) to O (log|O|) through k-d tree indexing of obstacle coordinates, worst-case performance may still degrade substantially when obstacle density becomes large relative to graph size. In such cases, extensive pruning can fragment the graph into disconnected regions, increasing failed search expansions and reducing the effectiveness of heuristic guidance. Although the algorithm often exhibits fast termination under these conditions, pathological cases may still approach the worst-case exploration behavior of graph search algorithms before infeasibility is determined.

5.3. Future Work

Future work will pursue several directions. Scalability to larger metropolitan networks, such as the full Kathmandu Valley including outer ring roads, will be investigated through further spatial indexing optimizations. The obstacle model will be extended to heterogeneous, time-varying obstacles to improve real-world fidelity. Integration with predictive traffic or hazard models could enable proactive rather than reactive navigation. Additionally, the framework's applicability to multi-agent and autonomous navigation systems will be explored, where multiple agents must simultaneously find constraint-safe paths in shared dynamic environments.

5.4. Conclusion

This research provides a validated, real-time framework for safety-critical path planning that balances efficiency, reliability, and verifiable safety. The hard exclusion mechanism offers a meaningful and demonstrable improvement over existing soft-penalty approaches by providing provable obstacle avoidance guarantees. The empirical evaluation on complex urban road networks establishes a foundation for next-generation intelligent transportation systems, with clear pathways for extension toward more dynamic, heterogeneous, and large-scale navigation scenarios.

References

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V. and Likhachev, L. (2016) 'Multi-Heuristic A*,' *The International Journal of Robotics Research*, 35(1–3), pp. 224–243. doi: 10.1177/0278364915594029.

Bentley, J.L. (1975) 'Multidimensional Binary Search Trees Used for Associative Searching,' *Communications of the ACM*, 18(9), pp. 509–517. doi: 10.1145/361002.361007.

Foeada, D., Ghifaria, A. and Kusu, M.B. (2021) 'A Systematic Literature Review of A* Pathfinding,' *Procedia Computer Science*, 179, pp. 507–514. doi: 10.1016/j.procs.2021.01.034.

Goldberg, A.V. and Harrelson, C. (2005) 'Computing the Shortest Path: A* Search Meets Graph Theory,' in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '05)*, pp. 156–165. Available at: https://dl.acm.org/doi/10.5555/1070432.1070455.

Isa, N., Mohamed, A. and Yusoff, M. (2015) 'Implementation of Dynamic Traffic Routing for Traffic Congestion: A Review,' in *Communications in Computer and Information Science: Soft Computing in Data Science (SCDS 2015)*, vol. 545. Singapore: Springer, pp. 174–186. doi: 10.1007/978-981-287-936-3_17.

Jerbi, M., Meraihi, R., Senouci, S.-M. and Ghamri-Doudane, Y. (2006) 'GyTAR: improved Greedy Traffic Aware Routing Protocol for Vehicular Ad Hoc Networks in City Environments,' in *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks (VANET '06)*, Los Angeles. doi: 10.1145/1161064.1161080.

Kaindl, H. and Kainz, G. (1997) 'Bidirectional Heuristic Search Reconsidered,' *Journal of Artificial Intelligence Research*, 7, pp. 283–317. doi: 10.1613/jair.460.

Li, J., Kang, F., Chen, C., Tong, S., Jia, Y., Zhang, C. and Wang, Y. (2023) 'The Improved A* Algorithm for Quadrotor UAVs under Forest Obstacle Avoidance Path Planning,' *Forests*, 14(3), p. 575. doi: 10.3390/app13074290.

Liu, L., Wang, B. and Xu, H. (2022) 'Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method,' *Mathematical Problems in Engineering*, 2022, p. 3694550. doi: 10.3390/electronics11223660.

Sabar, N.R., Bhaskar, A., Chung, E., Turky, A. and Song, A. (2019) 'A Self-adaptive Evolutionary Algorithm for Dynamic Vehicle Routing Problems with Traffic Congestion,' *IEEE Transactions on Evolutionary Computation*, 23(6), pp. 1001–1014. doi: 10.1016/j.swevo.2018.10.015.

Song, L., Su, Y., Dong, Z., Shen, W., Xiang, Z. and Mao, P. (2018) 'A two-level dynamic obstacle avoidance algorithm for unmanned surface vehicles,' *Ocean Engineering*, 170, pp. 351–360. doi: 10.1016/j.oceaneng.2018.10.008.

Wan, X., Ghazzai, H. and Massoud, Y. (2019) 'Real-Time Navigation in Urban Areas Using Mobile Crowd-Sourced Data,' in *Proceedings of the IEEE International Systems Conference (SYSCON'19)*, Orlando, FL, USA, April. doi: 10.1109/SYSCON.2019.8836809.

Yan, L., Wenbin, H. and Hu, S. (2019) 'SALA: A Self-Adaptive Learning Algorithm—Towards Efficient Dynamic Route Guidance in Urban Traffic Networks,' *Neural Processing Letters*, 50, pp. 77–101. doi: 10.1007/s11063-018-9870-0.

Zhang, Y., Li, B., Huo, T. and Liu, R. (2025) 'Research on dynamic obstacle avoidance method for robots by integrating and improving A* algorithm and DWA algorithm,' *Journal of System Simulation*, 37(6), pp. 1555–1564 (ahead of print). doi: 10.16182/j.issn1004731x.joss.24-0143.