Real Time Pathfinding with Heuristic Algorithms

Sai Durga Lakshmi SriLasya Addala
dept. of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
saddala@stevens.edu

Priyanshu Singh Bisen
dept. of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
psbisen@stevens.edu

Prateek Pravanjan
dept. of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
ppravanj@stevens.edu

***Abstract*—Dynamic pathfinding in real-time environments poses substantial challenges due to continuously changing obstacle positions and goal locations. This study conducts a comparative analysis of heuristic and greedy algorithms, including A*, Dijkstra's, Bidirectional A*, Breadth-First Search, Best-First Search, and Minimum Spanning Tree, to evaluate their effectiveness in dynamic pathfinding scenarios. The research focuses on how these algorithms dynamically adapt to environmental changes without relying on prior knowledge or precomputed paths. Key performance metrics such as time-per-planning (TPP), overall scores, Arrival rate and Accumulative success path are utilized to assess their responsiveness and efficiency. By systematically comparing heuristic and greedy approaches, this study aims to identify strengths, weaknesses, and trade-offs in real-time navigation.**

I. Introduction

Real-time navigation in dynamic and unpredictable environments is a critical challenge in autonomous systems, and interactive simulations. Dynamic environments introduce additional complexity due to the need for agents to adapt quickly to changes [1]. This is particularly true in scenarios involving moving obstacles, which can follow deterministic patterns, i.e., consistently moving obstacles, or exhibit unpredictable behaviors, i.e., randomly moving obstacles. In such settings, obstacles may appear, disappear, or move unpredictably, requiring agents to continuously adjust their paths to maintain efficiency and avoid collisions. Reactive re-planning emerges as a practical approach to address these challenges, relying on real-time path re-computation whenever environmental changes occur [2]. Unlike adaptive learning-based methods, reactive re-planning does not involve long-term memory or learning from prior experiences. That means the agent does not learn or adapt to obstacles but relies on reactive pathfinding, recomputing the path whenever the environment changes.

This study explores the performance of several pathfinding algorithms such as A*, Dijkstra's, Bidirectional A*, Breadth-First Search, Minimum Spanning Tree, and Best-First Search, in both CMO and RMO scenarios. Each algorithm brings unique characteristics to the table, impacting its suitability for different dynamic scenarios. For example, A* is known for its optimal and complete solutions when a proper heuristic is used, making it a strong candidate for environments where computational efficiency and path accuracy are both critical. Dijkstra's algorithm, although slower due to its exhaustive exploration of nodes, guarantees an optimal path in scenarios without heuristic guidance. Bidirectional A*, on the other hand, significantly reduces search time by simultaneously expanding nodes from the start and goal, making it well-suited for large environments.

Breadth-First Search (BFS) and Minimum Spanning Tree (MST) algorithms offer contrasting benefits. BFS systematically explores nodes layer by layer and is effective in finding the shortest path in unweighted graphs, but its lack of heuristic optimization can lead to inefficiency in larger, weighted environments. MST, while not a traditional pathfinding algorithm, can serve as a framework for navigating connected networks, especially when spanning connectivity is prioritized over direct paths. Best-First Search excels in environments with clear heuristics, prioritizing nodes closest to the goal and often achieving faster results, though it risks overlooking optimal paths if the heuristic is not well-tuned.

The evaluation focuses on key metrics such as Time Per Planning (TPP), score arrival rate, overall scores and accumulative success path to evaluate the performance of these algorithms [3]. By simulating a dynamic environment and analyzing results, this work aims to compare various contrasting algorithms in dynamic environments. This system aims to improve real-time navigation in unpredictable and complex environments.

II. Design and Algorithms

A. Design Overview

1) Environmental setup:

- Obstacles: Obstacles are represented as a collection of segments in a 2D grid space. These obstacles may either move predictably (CMO) or randomly (RMO), which affects the pathfinding logic.
- Pathfinding Agents: These are the robots or agents that compute and follow the paths in the environment. The agents are reactive, recomputing their paths when obstacles move or when the goal changes.

2) Environment Representation:

- Obstacle Tensor: The static positions of obstacles are stored in a 3D tensor representing the positional data of obstacles. Each obstacle consists of four edges, with start and end points in 2D space.

Shape: (O, 4, 2, 2)

$$O : \text{number of obstacles}$$
$$4 : \text{edges per obstacle}$$
$$2 \times 2 : \text{2D coordinates for start and end}$$

- Velocity Tensor: For dynamic obstacles, the velocity tensor stores the dynamic movement information. This helps in dynamically updating obstacle positions during pathfinding.
  Shape: (O, 1, 1, 2)

$$O : \text{number of obstacles}$$
$$1 : \text{singleton dimension (for broadcasting)}$$
$$2 : \text{velocity components (x and y)}$$

The extra singleton dimensions ensure efficient tensor broadcasting for dynamic updates. The last dimension represents the velocity components in the x and y directions.

3) Environment Types: The project evaluates performance in two types of dynamic environments:

CMO (Consistently Moving Obstacles)	**RMO (Randomly Moving Obstacles)**
Obstacles move consistently in a fixed direction.	Obstacles move with random velocity perturbations.
Velocities reverse only when hitting boundaries.	Velocities are randomized within limits at each iteration.
Highly predictable obstacle behavior.	Unpredictable obstacle movement.
Easier to handle due to deterministic trajectories.	More challenging due to random movement patterns.

4) Design Key Features:

- Ability to reuse previously computed results.
- Reduces computation time.
- Adaptability to changes in the graph.

B. Algorithms

Six path-finding algorithms are evaluated:

- **A*:** Balances exploration and path cost optimization.
- **Dijkstra:** Uniform cost search, guaranteed to find the shortest path.
- **Best First:** A greedy algorithm that prioritizes heuristic values.
- **Bi-directional A*:** Searches simultaneously from the start and the goal.
- **Breadth First Search (BFS):** Explores all nodes at the current depth before moving deeper.
- **Minimum Spanning Tree (MST):** Focuses on constructing a tree with minimal edge costs.

III. Implementation

The dynamic pathfinding system is designed to simulate and analyze the performance of different algorithms in a dynamic environment.

A. Obstacle Generation

Obstacles are preset and saved as part of the environment configuration.

This configuration ensures consistency across different simulation runs, allowing fair comparisons between pathfinding algorithms.

B. Environment Initialization

The environment is set up to simulate the conditions in which pathfinding algorithms will operate. This includes setting up the simulation, loading obstacles and direction vectors along with initializing algorithms with start position and a moving goal.

- Load obstacles and define their behavior (for example, fixed, controlled movement, or random movement).
- Initialize the algorithms by setting up the starting position and specifying a moving goal.
- Direction vectors are configured to guide moving obstacles and goals, ensuring dynamic behavior in the environment.

C. Pathfinding Execution

Pathfinding algorithms are evaluated in a controlled simulation environment to assess their performance under identical conditions.

- Batch Evaluation: All selected algorithms are executed, and their performance is evaluated during each simulation cycle.
- Play: Visualizes algorithms and collects metrics for comparison on iterations of simulations for averaging metrics in different environment instances.

D. Result Analysis

The final stage involves aggregating performance metrics and an exhaustive study on behavioral differences of the heuristic and greedy algorithms.

IV. Evaluation Metrics

A. Time Per Planning (TPP)

Measures the average computation time(in seconds) required for the algorithm to find a path at each planning step. It is calculated incrementally to reflect real-time performance.

Computed incrementally during dynamic planning:

$$\text{TPP} = \text{TPP} + \frac{\text{time}_{\text{elapsed}} - \text{TPP}}{c}$$

B. Score

The score evaluates the effectiveness of the algorithm in finding a valid and efficient path and it is tied to travel distance. It is computed as the sum of Euclidean distances between consecutive waypoints.

$$\text{distance} += \sqrt{(x_{\text{start}} - x_{\text{next}})^2 + (y_{\text{start}} - y_{\text{next}})^2}$$

C. Success Path Length

The average distance traveled by the agent for successful paths, specifically those with non-zero scores. This metric reflects the efficiency of the agent in reaching its destination when a path is found.

D. Arrival Rate

The fraction of runs in which the agent successfully reaches the target. This metric indicates the reliability of the pathfinding algorithm in achieving the goal, regardless of the path length.

V. Testing and Simulation Results

A. Maze

The image below showcases a maze simulation where the goal is to navigate from a start point to a destination while avoiding obstacles. In this visualization, the black boxes represent the obstacles scattered throughout the environment, preventing direct paths. The line shown in the image illustrates the path taken by the algorithm in an effort to navigate from the starting point to the goal.

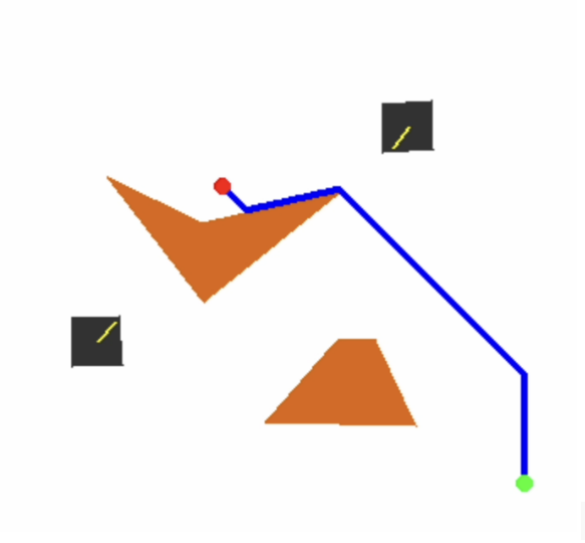

Fig. 1. Maze Simulation

B. Results

1) Success path length comparison(CMO and RMO): The graph shows the success path length comparision for all the algorithms we used. BestFirst, A*, and Bi-directional A* achieved high scores.

2) Time Per Planning vs Score(CMO and RMO): BestFirst, A* and Bi-Directional A* balance fast planning times with high-quality scores. Dijkstra's and MST take longer for planning but still produce acceptable solutions.

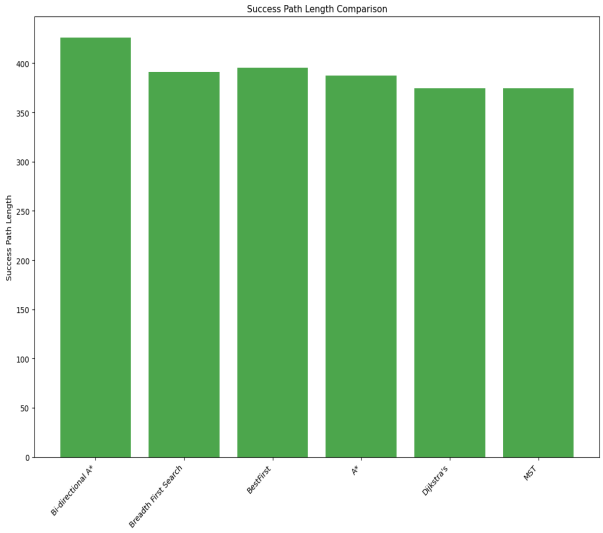

Fig. 2. SPL for CMO

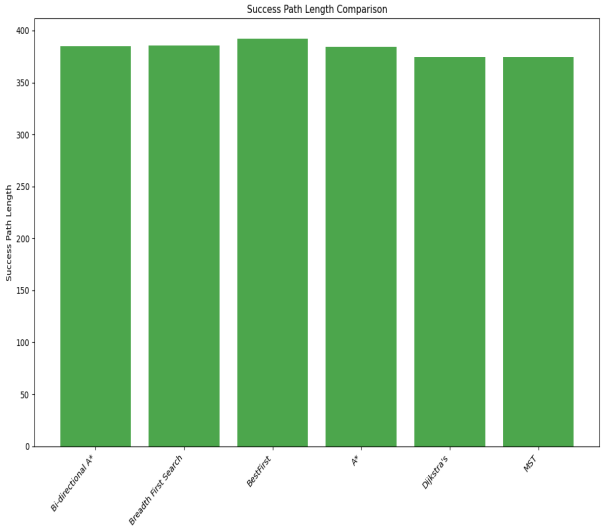

Fig. 3. SPL for RMO

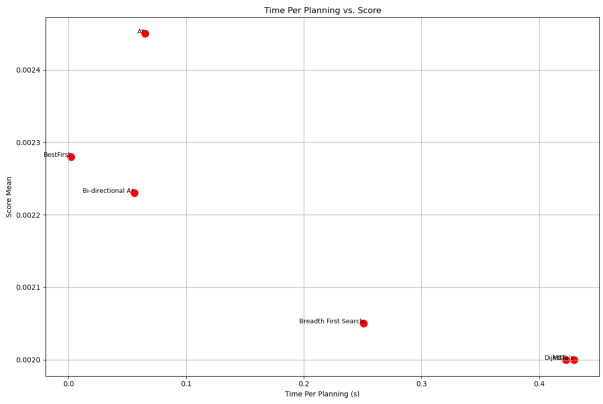

Fig. 4. TPP vs Score for CMO

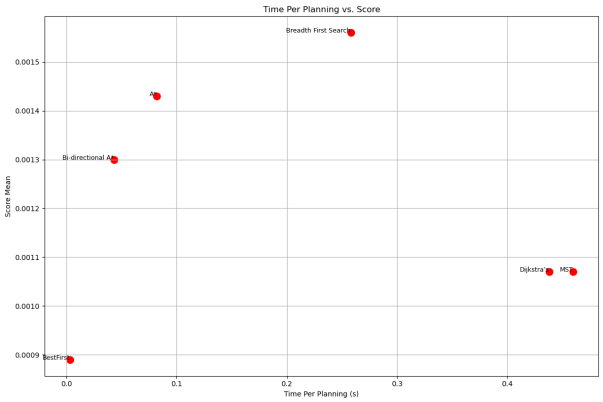

Fig. 5. TPP vs Score for RMO

VI. Analysis and Discussions

This study evaluates the performance of various pathfinding algorithms in two distinct dynamic environments: CMO (Consistently Moving Obstacles) and RMO (Randomly Moving Obstacles). The findings reveal that algorithm performance is highly influenced by the nature of the environment, highlighting critical trade-offs between reliability, computational efficiency, and adaptability.

In CMO, where obstacles move predictably, algorithms such as Bi-Directional A* and A* demonstrate the highest reliability, achieving a 95% arrival rate while maintaining strong pathfinding metrics and computational efficiency. Best First Search, while extremely fast, sacrifices reliability, making it suitable only for scenarios where speed is prioritized over path accuracy.

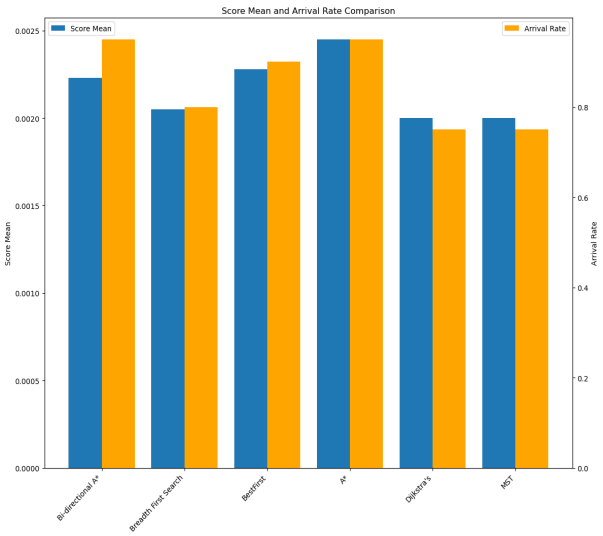

Fig. 6. Score Mean and Arrival Rate Comparison

In contrast, RMO presents a greater challenge due to the randomness of obstacle movements. None of the algorithms match their CMO performance, but BFS and A* adapts better than others, achieving arrival rates of 60% and 55%, respectively. Greedy algorithms, such as Best First Search, struggle significantly in RMO environments, underscoring their limitations in handling dynamic and unpredictable scenarios.

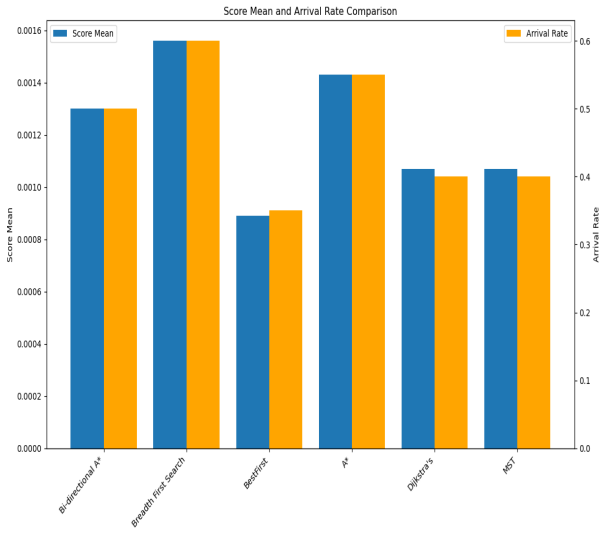

Fig. 7. Score Mean and Arrival Rate Comparison

A key trade-off observed is between path optimality and computational efficiency. Algorithms like Dijkstra and MST prioritize optimal paths but at the cost of high computational time, making them less suitable for real-time applications. Conversely, greedy approaches like Best First Search are computationally efficient but lack robustness in dynamic settings. Based on these observations, we recommend:

1) For predictable environments like CMO:

- Bi-Directional A* and A* are ideal choices due to their balance of reliability and efficiency.
- Best First Search can be considered for applications where computational speed is critical, albeit at the expense of reliability.

2) For unpredictable environments like RMO:

- Algorithms with better adaptability, such as BFS and A*, are more suitable despite their lower arrival rates.
- These insights offer a comprehensive understanding of the strengths and limitations of pathfinding algorithms under varying environmental dynamics, providing a basis for selecting appropriate algorithms for specific real-world applications.

These insights offer a comprehensive understanding of the strengths and limitations of pathfinding algorithms under varying environmental dynamics, providing a basis for selecting appropriate algorithms for specific real-world applications.

VII. Conclusion

A* and Bidirectional A*, being an adaptive algorithm, showed to be most suitable for unpredictable environments despite having lower arrival rates and is preferred for predictable

environments as well for being computationally efficient. Algorithms like Dijkstra and MST prioritize optimality at the cost of computational efficiency. Greedy approaches like Best First are fast but unreliable in dynamic settings.

REFERENCES

[1] A. Janis and A. Bade, "Path planning algorithm in complex environment: A survey," *Transactions on Science and Technology*, vol. 3, pp. 31–40, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:39394361

[2] K. Ota, D. Jha, T. Onishi, A. Kanezaki, Y. Yoshiyasu, Y. Sasaki, T. Mariyama, and D. Nikovski, "Deep reactive planning in dynamic environments," 10 2020.

[3] J. Xin, J. Kim, S. Chu, and N. Li, "Okayplan: Obstacle kinematics augmented dynamic real-time path planning via particle swarm optimization," *Ocean Engineering*, vol. 303, p. 117841, 07 2024.