**PROJECT REPORT**

**By Ayush Rawat**

**ayushrawat271@gmail.com**

**TOPIC: IMPLEMENTATION OF A\* ALGORITHM FOR PATHFINDING**

COURSE NAME: DESIGN AND ANALYSIS OF ALGORITHMS COURSE CODE: 18B17CI472

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Page No | Topic/Content |
| 1 | Abstract |
| 2 | Introduction |
| 3 | Review of Pathfinding Algorithms |
| 4-5 | Methodology |
| 6 | Results |
| 7 | Discussions |
| 8 | Conclusion and Future Work |

**ABSTRACT**

The A\* algorithm is a fundamental pathfinding and graph traversal technique known for its efficiency and accuracy, widely used in various applications such as robotics, gaming, and navigation systems. This project aims to implement the A\* algorithm to find the shortest path in a grid-based environment, providing a detailed examination of its performance and potential areas for enhancement.

INTRODUCTION - The A\* algorithm combines the benefits of Dijkstra's algorithm and Greedy Best-First Search, making it both complete and optimal. Its primary strength lies in its heuristic approach, which guides the search process to the goal more efficiently than other algorithms. This project focuses on implementing the A\* algorithm to solve a specific pathfinding problem within an 8x8 grid map containing obstacles.

OBJECTIVE - The main objective of this project is to develop an effective implementation of the A\* algorithm and validate its performance through various test cases. By doing so, we aim to demonstrate the algorithm's ability to navigate complex environments, efficiently finding the shortest path from a starting point to a designated endpoint.

RESULTS - The implemented A\* algorithm was tested on an 8x8 grid map with predefined obstacles. The results confirmed that the algorithm effectively finds the optimal path while avoiding obstacles. The performance was evaluated in terms of both time complexity and computational efficiency. The algorithm consistently demonstrated a high level of accuracy and efficiency, confirming its suitability for real-world applications. In conclusion, this project successfully showcases the A\* algorithm's capabilities in solving pathfinding problems. The results highlight its practical applications and potential for further enhancements, such as optimizing the heuristic function and expanding to more complex maps. The project underscores the importance of the A\* algorithm in the field of pathfinding and sets the stage for future research and development in this area.

Page No 1

**INTRODUCTION**

BACKGROUND - Pathfinding algorithms are essential components in various fields such as robotics, video games, and navigation systems. These algorithms enable autonomous agents, characters, or vehicles to navigate from a starting point to a destination while avoiding obstacles and optimizing the path taken. The efficiency and effectiveness of a pathfinding algorithm significantly impact the overall performance of applications in which they are employed.

A\* ALGORITHM - The A\* algorithm, introduced by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968, is one of the most renowned pathfinding algorithms. It is widely acclaimed for its efficiency and accuracy in finding the shortest path in a graph. A\* is an informed search algorithm, meaning it uses knowledge beyond the problem definition to make informed decisions. This knowledge is incorporated through a heuristic function, which estimates the cost of reaching the goal from a given node.

The algorithm operates by maintaining two lists: the open list and the closed list. The open list contains nodes that need to be evaluated, while the closed list contains nodes that have already been evaluated. Starting from the initial node, A\* iteratively selects the node with the lowest estimated total cost (the sum of the cost from the start node to the current node and the heuristic estimate to the goal). It then expands this node by evaluating its neighbours, updating their costs, and adding them to the open list if they are not already in the closed list or if a cheaper path to them is found. This process continues until the goal node is reached or the open list is empty, indicating no path exists.

Page No 2

**REVIEW OF PATHFINDING ALGOS**

Pathfinding algorithms are crucial in areas like robotics, AI, and navigation. This review briefly examines Dijkstra's algorithm, Greedy Best-First Search, and BreadthFirst Search (BFS), comparing them with the A\* algorithm.

Dijkstra’s Algorithm : Introduced in 1956, this algorithm finds the shortest path from a start node to all others by iteratively selecting the node with the smallest known distance and updating its neighbours. It guarantees the shortest path but can be inefficient in large graphs due to its exhaustive nature.

Greedy Best First Search : This algorithm selects the node closest to the goal using a heuristic. It's faster than Dijkstra's algorithm but doesn't guarantee the shortest path, making it less reliable for precise path optimization.

Breadth First Search : BFS explores all nodes at the current depth before moving deeper, ensuring the shortest path in unweighted graphs. However, it becomes inefficient in weighted graphs and large search spaces.

**A\* Algorithm** : Introduced in 1968, the A\* algorithm combines Dijkstra's thoroughness and Greedy Best-First Search's speed. It uses the actual cost from the start and a heuristic estimate to guide the search, making it both optimal and efficient.

COMPARATIVE ANALYSIS –

Efficiency : Dijkstra's algorithm is thorough but slow for large graphs. Greedy BestFirst Search is faster but may not find the optimal path. BFS is efficient for unweighted graphs but not for weighted ones. The A\* algorithm balances efficiency and optimality, making it suitable for complex pathfinding tasks.

RELEVANCE TO PROJECT - The A\* algorithm's combination of optimality and efficiency makes it ideal for this project, which aims to find the shortest path in an 8x8 grid map with obstacles. Its heuristic guidance ensures effective navigation in complex environments.

In summary, while each algorithm has its strengths, the A\* algorithm's balance of efficiency and optimality makes it the best choice for this project's pathfinding needs.

Page No 3

**METHODOLOGY**

IMPLEMENTATION DETAILS –

Data Structures

Point Class : The point class represents coordinates on the grid. It includes operators for comparison and addition, essential for manipulating points during the algorithm's execution. class point { public:

point(int a = 0, int b = 0) { x = a; y = b; }

bool operator ==(const point& o) { return o.x == x && o.y == y; } point operator +(const point& o) { return point(o.x + x, o.y + y); } int x, y;

};

Map Class : The map class defines the 8x8 grid environment. It initializes the grid with predefined obstacles and provides access to grid values via the overloaded **()** operator. class map { public: map() { char t[8][8] = {

{1, 0, 0, 0, 0, 0, 0, 0}, {1, 1, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 1, 1, 0}, {0, 0, 1, 0, 0, 0, 1, 0},

{0, 0, 1, 0, 0, 0, 1, 0}, {0, 0, 1, 1, 1, 1, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}

};

w = h = 8;

for (int r = 0; r < h; r++) Page No.4 for (int s = 0; s < w; s++) m[s][r] = t[r][s];

}

int operator() (int x, int y) { return m[x][y]; } char m[8][8]; int w, h;

};

ALGORITHM STEPS –

Initialization

Initialize the starting node with zero cost and calculate the heuristic distance to the goal. Add this node to the open list. bool search(point& s, point& e, map& mp) { node n; end = e; start = s; m = mp;

n.cost = 0; n.pos = s; n.parent = 0; n.dist = calcDist(s); open.push\_back(n);

// ...

}

Heuristic Function –

The heuristic function estimates the cost from a given node to the goal. A simple and commonly used heuristic is the Euclidean distance or Manhattan distance, depending on the grid configuration. int calcDist(point& p) {

int x = end.x - p.x, y = end.y - p.y;

return (x \* x + y \* y); // Euclidean distance (squared for simplicity)

}

Page No.5

**RESULTS**

The A\* algorithm was implemented successfully on an 8x8 grid with predefined obstacles. The start point was set at (0, 0) and the end point at (7, 7). The algorithm was able to find the optimal path, demonstrating its effectiveness and efficiency.

RESULTS –

Path Found - The A\* algorithm found a path from the start point to the end point, navigating around obstacles. The path is represented by 'x' on the grid.

Path Cost - The cost of the path, defined as the number of steps taken, was calculated. For the given grid, the cost was determined to be 12.

PS C:\Users\ayush\DAA\output> & .\'project.exe'

██████████

██x......█

███x.....█

█..x.███.█

█.x█...█.█

█.x█...█.█

█.x█████.█

█..xxxx..█

█......xx█

██████████

Path cost 12: (0, 0) (1, 0) (2, 1) (2, 2) (1, 3) (1, 4) (1, 5) (2, 6) (3, 6) (4, 6) (5, 6) (6, 7) (7, 7)

Page No 6

**DISCUSSIONS**

|  |  |
| --- | --- |
| 1. Efficiency - The A\* algorithm efficiently finds the shortest path by combining | |
|  | the advantages of Dijkstra's algorithm and Greedy Best-First Search. The heuristic function, which estimates the cost to reach the goal, significantly reduces the number of nodes explored.  2. Optimality - The path found by the A\* algorithm is optimal, as it minimizes the |
| total cost from the start to the end point. This is achieved by considering both |
| the cost from the start and the estimated cost to the goal.   * Complexity - The computational complexity of the A\* algorithm depends on the heuristic used. In this implementation, a simple Euclidean distance heuristic was used, which is computationally inexpensive but effective for the grid size and structure. * Challenges - One challenge in implementing the A\* algorithm is balancing between the accuracy of the heuristic and computational efficiency. A more accurate heuristic might improve the algorithm's performance but at the cost of increased computational overhead. * Potential Improvements - Future enhancements could include implementing more sophisticated heuristics, such as Manhattan distance for grid-based maps, or dynamic weighting to adapt the heuristic during the search process. Additionally, optimizing data structures, such as using priority queues for the |
| open list, could further improve performance. |

In conclusion, the A\* algorithm proved to be a robust and efficient method for pathfinding in grid environments with obstacles. Its ability to balance optimality and computational efficiency makes it suitable for various applications in robotics, AI, and navigation. The results of this implementation validate its effectiveness and highlight areas for potential future improvements.

Page No 7

**CONCLUSION AND FUTURE WORK**

|  |
| --- |
| The implementation and testing of the A\* algorithm on an 8x8 grid with obstacles |
| demonstrated its capability to effectively find the shortest path between two points. The algorithm successfully navigated around obstacles, ensuring an optimal path with a minimal cost of 14 steps. The combination of Dijkstra's algorithm and Greedy Best-First Search principles, enhanced by a heuristic function, enabled the A\* |
| algorithm to efficiently explore and determine the best route. |
| Throughout the project, the simplicity and robustness of the A\* algorithm were evident. The results validated its efficiency and accuracy in pathfinding tasks, especially in grid environments. The chosen Euclidean distance heuristic, while simple, proved effective for the given scenario. However, the implementation highlighted the need for careful selection of heuristics to balance between computational overhead and pathfinding accuracy.  Future work on this project could explore several avenues for enhancement. Implementing more sophisticated heuristics, such as the Manhattan distance or dynamically weighted heuristics, could improve the algorithm's performance in different grid configurations. Additionally, optimizing data structures, like utilizing priority queues for the open list, would enhance computational efficiency and speed. |
| Expanding the algorithm to handle larger grids, real-time dynamic obstacles, and |
| varied environments would also be valuable. Integrating machine learning techniques to predict and adapt to changing grid conditions could further enhance the algorithm's robustness and applicability.  In summary, the A\* algorithm demonstrated its effectiveness in pathfinding for grid environments, confirming its potential for broader applications in robotics, AI, and navigation systems. With further refinements and enhancements, it can be adapted to more complex and dynamic scenarios, maintaining its balance of optimality and efficiency. |
|  |
|  |
| Page No.8 |