

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | **2117240030008** |
| **NAME** | Anushiya p |
| **PROJECT TITLE** | Route optimization in a city map using uniform cost search |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**AI MINI PROJECT REPORT**

Route Optimization in a City Using Uniform Cost Search

INTRODUCTION

Overview of Artificial Intelligence:

Artificial Intelligence (AI) is the field of computer science that enables machines to perform tasks that typically require human intelligence — such as reasoning, learning, and decision-making. AI techniques like search algorithms, machine learning, and optimization help solve real-world problems efficiently.

Project Introduction:

In urban environments, route optimization is an important problem where one must find the shortest or least-cost path between two locations. This project applies the Uniform Cost Search (UCS) algorithm, a classical AI search technique, to solve this problem. UCS ensures that the path with the lowest cumulative cost is always chosen first.

Why It Matters:

Efficient route planning reduces travel time, fuel consumption, and traffic congestion — benefiting transportation, logistics, and delivery systems.

Project Aim:

To design and implement an AI-based route optimization model using Uniform Cost Search to find the optimal path between two points in a city network.

PROBLEM STATEMENT

To find the most cost-efficient path between two points in a city where each road segment has a different distance or travel cost. The system should automatically determine the shortest possible route using AI search techniques.

GOAL

The goal of this project is to:

Determine the shortest and least costly route in a city map.

Reduce computational time and guarantee optimality.

Demonstrate how Uniform Cost Search can be applied in real-world navigation systems.

THEORETICAL BACKGROUND

1. About the Problem:

Route optimization involves identifying a path between two nodes in a weighted graph such that the total cost (e.g., distance or time) is minimized.

1. About the Algorithm (UCS):

Uniform Cost Search is a best-first search algorithm that explores the least-cost node first. It uses a priority queue to ensure that the next expanded node always has the lowest cumulative cost.

1. Literature Survey:

Dijkstra’s algorithm and A\* search are similar techniques for finding shortest paths.

UCS guarantees an optimal solution like Dijkstra but is simpler when heuristic data is unavailable.

1. Justification for Choosing UCS:

UCS is selected because:

It always finds the optimal path if costs are positive.

It is complete and suitable for weighted graphs.

It requires no heuristic function, making it ideal for simple city route maps.

ALGORITHM EXPLANATION WITH EXAMPLE

Algorithm Steps:

1. Initialize the start node with cost = 0.

2. Store paths in a priority queue based on total cost.

3. Expand the node with the lowest cost.

4. For each neighbor, calculate the new path cost and update if cheaper.

5. Stop when the goal node is reached.

Example:

Consider the city graph:

A → B (3), A → C (1)

B → E (1), C → B (1), C → D (1), D → E (5)

The UCS explores paths in increasing order of cost.

It finds A → C → B → E as the optimal path with total cost 3.

IMPLEMENTATION AND CODE

Import heapq

Def uniform\_cost\_search(graph, start, goal):

Queue = [(0, start, [start])] # (cost, current\_node, path)

Visited = set()

While queue:

Cost, node, path = heapq.heappop(queue)

If node == goal:

Return cost, path

If node not in visited:

Visited.add(node)

For neighbor, distance in graph[node]:

Heapq.heappush(queue, (cost + distance, neighbor, path + [neighbor]))

Return float(“inf”), []

# Example graph

City\_graph = {

‘A’: [(‘B’, 3), (‘C’, 1)],

‘B’: [(‘D’, 3), (‘E’, 1)],

‘C’: [(‘D’, 1), (‘B’, 1)],

‘D’: [(‘E’, 5)],

‘E’: []

}

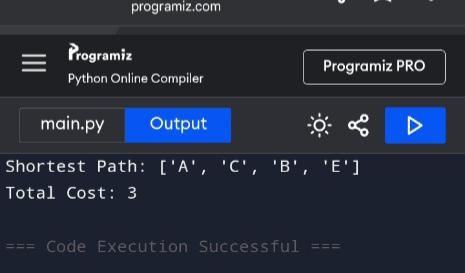
Start, goal = ‘A’, ‘E’

Cost, path = uniform\_cost\_search(city\_graph, start, goal)

Print(“Shortest Path:”, path)

Print(“Total Cost:”, cost)

OUTPUT

Output Screen:

Explanation:

From A, UCS first visits C (lowest cost = 1).

Then it expands C → B (cost = 2).

Finally, B → E adds cost 1, giving a total of 3.

Hus, A → C → B → E is the shortest route with total cost 3.

RESULTS AND FUTURE ENHANCEMENT

Results:

The Uniform Cost Search algorithm successfully computed the shortest path in the given city network with guaranteed optimality.

Future Enhancements:

Integrate real-time traffic data to make the model dynamic.

Implement A\* algorithm for faster performance with heuristics.

Build a GUI for user-friendly route visualization.

GitHub Link of the Project and Report:

(You can upload your code and report to GitHub and paste the link here, e.g.)

👉 <https://github.com/yourusername/route-optimization-ucs>

REFERENCES:

1. Russell, S. & Norvig, P. Artificial Intelligence: A Modern Approach, Pearson.

2. GeeksforGeeks – “Uniform Cost Search in Artificial Intelligence.”

3. TutorialsPoint – “Search Algorithms in AI.”

4. Python Official Documentation – heapq module.

5. Research Paper – Optimization of Shortest Path Algorithms for Smart Cities, IEEE, 2022.