UPES School of Computer Science & Engineering

# TRAVEL OPTIMIZER: A COMPARATIVE STUDY TO FIND THE OPTIMAL PATH

ISHA, JAHANVI SETHI, SAUMYA GUUPTA
MENTORED BY: DR SACHIN CHAUDHARY

UPES UNIVERSITY OF THE FUTURE

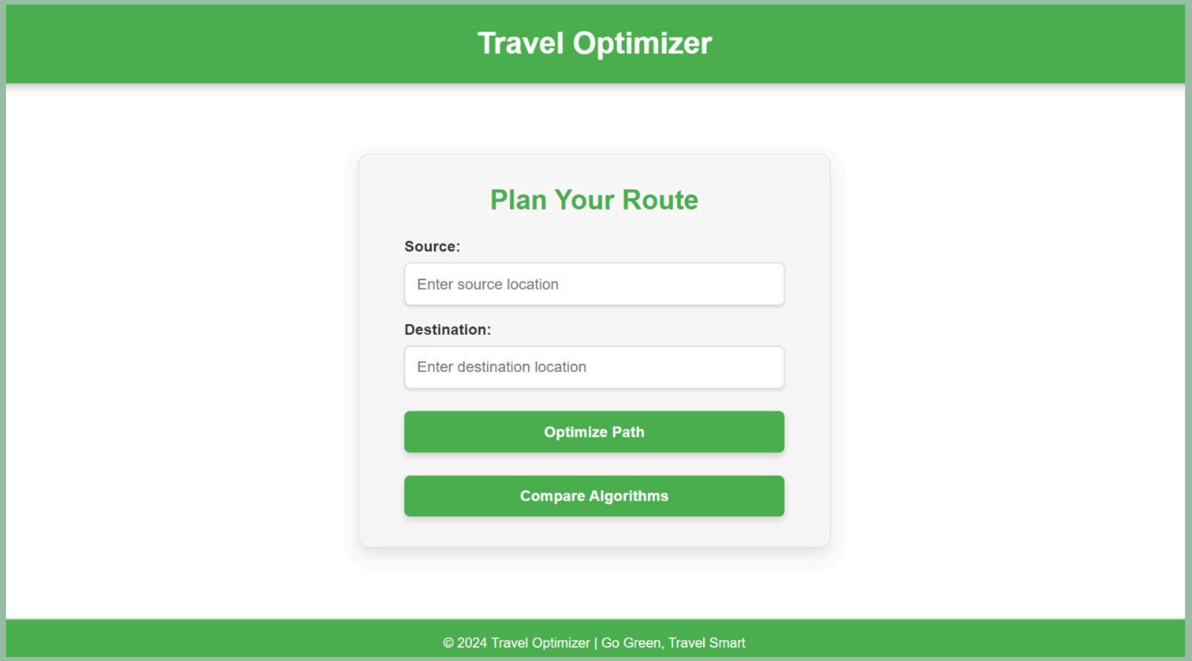
## ABSTRACT

The Travel Optimizer project addresses the challenge of finding the most efficient route between multiple destinations, including a return to the starting point. By allowing users to input only the starting and ending locations, the system automatically optimizes the path across all intermediate cities. Leveraging pathfinding algorithms, the project aims to minimize travel time, even as the number of destinations increases. Its scalable design harnesses advanced pathfinding algorithms to provide precision and efficiency, making it an indispensable tool for travelers and logistics alike.

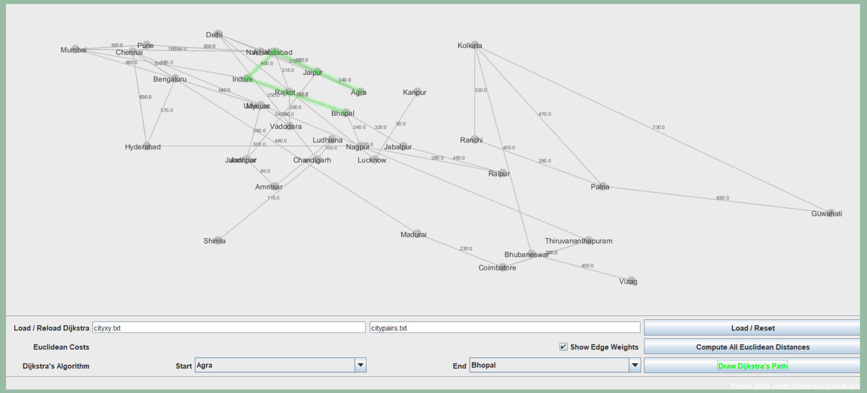
## INTRODUCTION

The Travel Optimizer project focuses on finding the most efficient routes for journeys involving multiple destinations. By analyzing and comparing different pathfinding algorithms, the system ensures optimal travel in terms of time, or distance,.

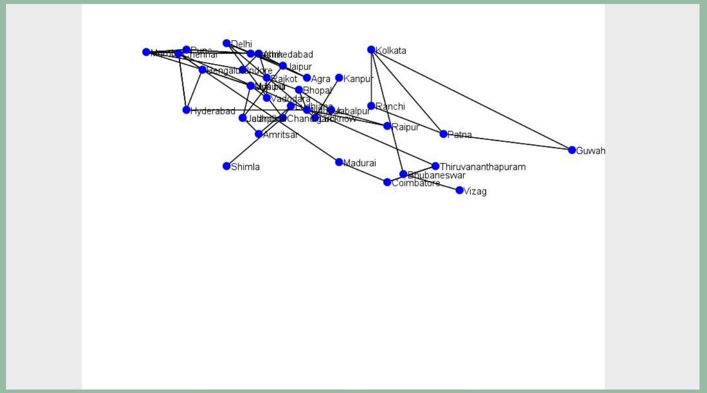
## OUTPUT



Dijkstra's Algorithm

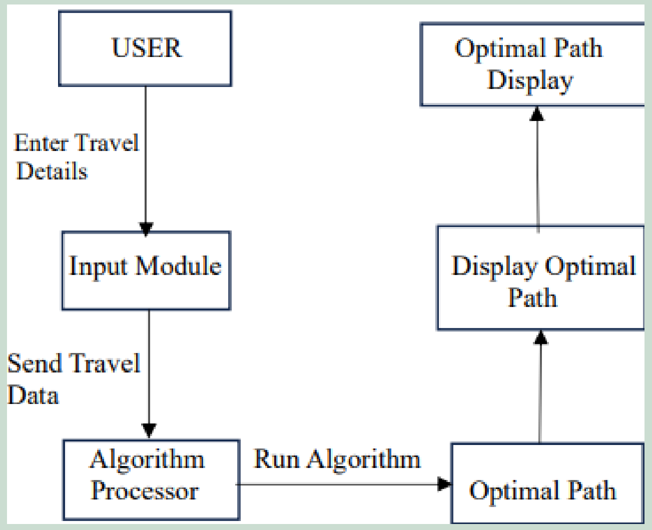


A* Algorithm



## METHODOLOGY

1.Defining the problem: Identification of the optimal path between multiple locations to minimize travel time and distance.
Deliverable: A clear problem statement that describes the objective
2.Data Set Collection: Collection or creation of graph representing locations and possible travel paths.
Deliverable: A dataset or graph structure representing locations and paths.
3. Algorithm Selection: Selection of pathfinding algorithms for comparison.
Deliverable: Algorithm Selection Report providing a comparative analysis of the algorithms considered, including their strengths, limitations, and suitability for specific travel conditions.
4. Algorithm Implementation



## RESULT/FINDINGS

Dijkstra's Algorithm:
- Performs best on dense graphs with non-negative weights, providing guaranteed shortest paths.
- High memory usage due to its reliance on a priority queue and the storage of all nodes in the graph.
- Less effective on large, sparse graphs or when the goal is localized pathfinding.
- Lacks flexibility for targeted search, as it explores all possible paths indiscriminately.

A Algorithm:*
- Excels with accurate heuristics in spatial graphs, especially in applications like navigation and robotics.
- Requires careful design of the heuristic function to avoid increased computational overhead.
- Performs poorly or equivalently to Dijkstra's in the absence of a well-informed heuristic.

## CONCLUSION

Each algorithm has its unique strengths and is better suited for certain situations. Dijkstra's algorithm works best in stable environments where all the weights are non-negative, providing accurate results for dense and static networks. On the other hand, the A* algorithm is ideal for applications involving movement or navigation, especially when a good heuristic is available to guide the search efficiently. Choosing the right algorithm depends on the needs of the specific task, such as the size of the graph, the type of data, and the importance of speed or accuracy. Understanding these factors helps in making the best decision for optimal performance.