Habib University
CS 351 - Artificial Intelligence

Project Report, Fall 2024

A Comparative Analysis of Search Algorithms for Inter-City Pathfinding Optimization in Pakistan

Mohammed Haider Abbas
Dhanani School of Science and Engineering
Habib University
ma06418@st.habib.edu.pk

Ali Zain Sardar
Dhanani School of Science and Engineering
Habib University
as06998@st.habib.edu.pk

***Abstract*—This paper talks about the comparative analysis of search algorithms for determining the shortest path between cities in Pakistan. Our study primarily revolves around the implementation and efficiency assessment of renowned algorithms like A*, Dijkstra's and Best First Algorithm. In pursuit of a practical understanding, we have constructed a bidirectional weighted graph representing the intricate network of cities, calculating distances using real geographical data. Our approach involved a through application of each algorithm to this graph which was followed by capturing their performance metrics in order to perform empirical analysis for comparison. This exploration not only highlights the strengths and limitations of each algorithm but also lays a foundation for future advancements in the application of AI techniques in urban planning and logistics. The insights gained from this study have significant implications for optimizing route planning in rapidly urbanizing regions, aiming to contribute to the growing body of knowledge in AI applications for real world challenges.**

Keywords: A* Search Algorithm, Pathfinding, Inter-City Navigation, Route Planning, Graph Theory, Optimization

I. Introduction

The rise of artificial intelligence has revolutionized fields like geographic information systems and urban planning by offering innovative solutions to complex problems. Search algorithms like A*, Dijkstra's & Best First Search, play a crucial role in optimizing routes and managing traffic flow world wide, specially in countries like Pakistan where there is a lot of room for improvement in route planning, specifically inter-city route planning and optimization. This report focuses on the application of these algorithms in addressing the pressing need for efficient route mapping in the rapidly growing urban landscape of Pakistan.

For this study, we have constructed a bidirectional weighted graph using real geographical data to calculate distances representing the network of Pakistan's top 83 cities. The selected search algorithms are evaluated for their performance, offering insights into their practical applications and efficiency. The aim is to not only compare the algorithms in terms of accuracy and efficiency but also to explore their potential implications for inter city route planning in Pakistan.

II. Background Study

Before analyzing the approach and the results, it is crucial to understand the theoretical foundations of the chosen algorithms: A*, Dijkstra' and Best First Search. These algorithms share the common goal of finding a path between two nodes in a graph, yet they are very much distinct and demonstrate varying efficiencies based on the characteristics of the graph. In this background study, we offer an overview of each algorithm, establishing the foundation for the subsequent comparative analysis.

A. A Search Algorithm*

The A* algorithm stands out for its use of heuristics to determine the shortest path in a graph more efficiently than typical shortest path algorithms. It combines features of Dijkstra's algorithm and the Greedy Best-First-Search, striking a balance between exploring the most promising routes and ensuring to find an optimal path. A* is particularly effective in applications where an estimation of the distance to the goal can be calculated, a feature that comes in handy in geographical route planning. The efficiency of A* relies on its heuristic function, typically a straight-line distance in geographical applications. This heuristic guides the search, enabling A* to explore fewer paths than Dijkstra's while still guaranteeing the shortest path, under the condition that the heuristic is admissible (it never overestimates the real cost to reach the destination).

B. Dijkstra's Algorithm

Dijkstra's algorithm is a foundational algorithm in the field of computer science, known for its effectiveness in finding the

shortest path in a graph. It systematically expands outward from the source, considering all possible paths and continuously updating the shortest path. This algorithm is particularly well suited for applications like road networks where the graph is dense, and paths are numerous. The primary strength of Dijkstra's algorithm is its guarantee of finding the shortest path in any graph with non-negative weights. However, its exhaustive nature can lead to inefficiencies, especially in larger graphs where the number of vertices and edges to consider grows significantly.

C. Best First Search (BestFS)

Best-First Search is a graph traversal algorithm that selects the most promising node for exploration based on a heuristic evaluation. Unlike Breadth-First Search (BFS) and Depth-First Search (DFS), BestFS focuses on exploring nodes that are estimated to be closer to the goal, as determined by a heuristic function. The algorithm maintains a priority queue where nodes are ordered based on their heuristic values. The heuristic function provides an estimate of the cost or distance from the current node to the goal. BestFS selects the node with the lowest heuristic value from the priority queue for exploration, making it a greedy algorithm. Best-First Search is a heuristic driven algorithm that explores nodes based on their estimated proximity to the goal. It is often employed when there is additional information available to guide the search efficiently, making it a versatile choice for various problem-solving applications.

III. Experiments/Analysis Conducted

A. Code Base

The complete code and all associated files for our project can be found at the following GitHub repository:
https://github.com/haider-06418/A-Star-Intercity-Optimization

B. Tools

- Python
- VS Code

C. Problem Formulation

1) Objective: To the analyze the performance of the search algorithms on the graph consisting of Pakistani cities.

2) Data Source: For this research, we needed to create a graph as exact and precise as we could for better results. Therefore, we manually listed 83 cities in Pakistan along with their exact coordinates. [1] From this data which we gathered, we calculated the distance between each city with one another based on the Haversine formula [2] which is used to find the great circle distance between two points based on coordinates, which is the shortest distance over the Earth's surface. This graph is then saved to a .JSON file so that we can access and use it for running the algorithms anytime. This is also implemented in Python in a way that the graph creation process is Country neutral, the function takes a .JSON file with list of city names and coordinates and makes a graph from it, therefore this can also be used for any country or geographical location world wide.

D. Algorithm Implementation

All the 3 algorithms take the following as input:

- A weighted & bidirectional Graph of Cities (Adjacency List)
- Current Node
- Goal Node

And return the following as output:

- Path of traversing
- Cost of traversing
- Number of nodes explored

Following is a brief description of our implementation of each of the algorithm:

1) A Algorithm:* [3] In our implementation of the A* algorithm, the `heuristic` function estimates the cost from the current node to the goal node by considering the smallest distance among available edges. The main function, `a_star_search`, utilizes a priority queue to explore nodes in order of increasing total cost, which is the sum of the cost so far and the heuristic estimate. The algorithm maintains data structures to keep track of the path, cost, and number of nodes explored. It iteratively selects the node with the lowest total cost, updates the cost and path information for neighboring nodes if a better path is found, and continues until the goal node is reached.

2) Dijkstra's Algorithm: [4] The implementation of Dijkstra's algorithm aims to find the shortest path from the starting node to the goal node. It employs a priority queue to explore nodes in order of increasing cumulative cost. The algorithm maintains data structures, including the `came_from` dictionary to track the predecessors of each node and the `cost_so_far` dictionary to store the cost incurred so far for each node. The function iteratively selects the node with the lowest cumulative cost, updates the cost and predecessor information for neighboring nodes if a better path is found, and continues until the goal node is reached.

3) Best First Search (BestFS): [5] The Best-First Search algorithm, also known as BestFS, is implemented to find the optimal path from the starting node to the goal node in a graph. This algorithm utilizes a priority queue to explore nodes based on their heuristic values, which estimate the cost from the current node to the goal node. The `heuristic` function calculates the heuristic value, considering the smallest distance among available edges. The main function, `best_first_search`, iteratively selects the node with the lowest heuristic value, updates the cost and path information for neighboring nodes if a better path is found, and continues until the goal node is reached.

E. Performance Metrics

The following performance metrics have been used for analysis of the algorithms:

- Time taken for execution (in seconds)
- Cost of traversing
- Number of nodes explored

F. Experimental Setup

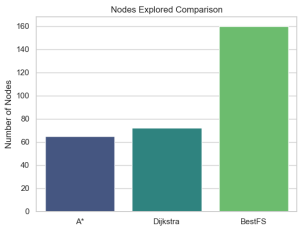
We have conducted many runs with arbitrary start and end nodes. Based on the categorizes of our findings, let's take the following test cases:

1) **Test Case 1**:
   - Graph = *graphs/graph_updated.json*
   - Current Node = *"Loralai"*
   - Goal Node = *"Jamshoro"*
2) **Test Case 2**:
   - Graph = *graphs/graph_updated.json*
   - Current Node = *"Jhelum"*
   - Goal Node = *"Gwadar"*
3) **Test Case 3**:
   - Graph = *graphs/graph_updated.json*
   - Current Node = *"Karachi"*
   - Goal Node = *"Islamabad"*

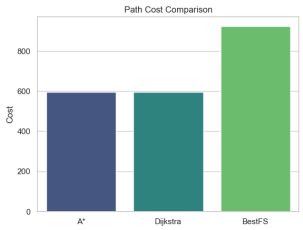
IV. Results

Based on our findings, we have categorized the results in to 3 categories. Each of test case corresponds to a category.

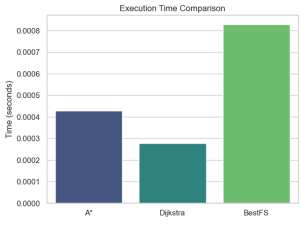
Test case 1 corresponds to category 1, and so on. Following are the results for Test case 1: If we analyze the results for



((a)) Nodes Explored Comparison



((b)) Path Cost Comparison

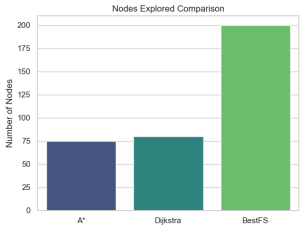


((c)) Execution Time Comparison

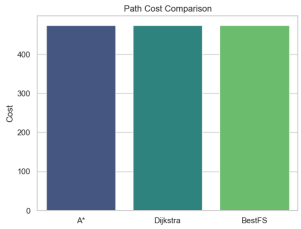
Fig. 1: Test Case 1 Results

test case 1 which represents the first category, we can see that BestFS was the highest in all 3 metrics: Most number of nodes explored, most costly path and most time taken for execution. This was the same with all the tests being done for this category. It can also be seen that A* explores the least number of nodes amongst the three which proves its optimal nature.

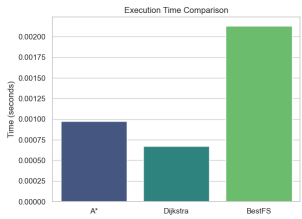
Following are the results for Test case 2: This category



((a)) Nodes Explored Comparison



((b)) Path Cost Comparison



((c)) Execution Time Comparison

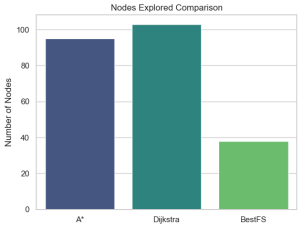
Fig. 2: Test Case 2 Results

redeems the performance of BestFS while showing the optimal nature of A*. Unlike category 1, many tests proved that BestFS can also give the same cost like A* and Djkistra. Although the number of nodes explored and time taken for execution were similar to category 1 results, here the cost is optimal meaning that BestFS has the capability to find the optimal path.

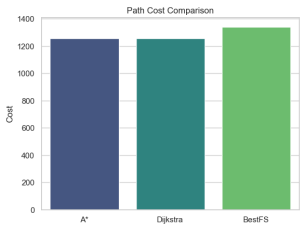
Following are the results for Test case 3 (Figure in Next Page): The third test case represents a corner case which was very rare but we found it therefore it deserved a category of it's own. From the results, we can see that the number of nodes explored by BestFS was significantly lower than A* and Djkisra and yet it gave almost the same cost (slightly greater than A* and Djkisra). Also for this case the roles in execution time were also reversed with A* taking the most time and BestFS taking less than half of that time. This behaviour was not present for the overwhelmingly cases but was present nonetheless.

Overall, all the results proved that A* is the most optimal.

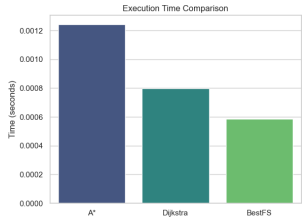
The comprehensive study and empirical analysis of A*, Dijkstra's, and BestFS algorithms, applied to the network of Pakistani cities have yielded insightful findings about the applicability and efficiency of these algorithms in real-world



((a)) Nodes Explored Comparison



((b)) Path Cost Comparison



((c)) Execution Time Comparison

Fig. 3: Test Case 3 Results

scenarios. This exploration has not only deepened our understanding of these algorithms but also highlighted their potential and limitations in the context of urban planning and intercity pathfinding. We can extract the following key findings from our analysis:

- A* algorithm exhibited remarkable efficiency in scenarios where an accurate heuristic was applicable. Its ability to significantly reduce the search space without compromising the accuracy of the shortest path proved advantageous in dense urban networks like those of Pakistan's cities.
- Dijkstra's Algorithm demonstrated its robustness in consistently finding the shortest path. Its exhaustive approach, while computationally intensive, was reliable, making it a dependable choice for comprehensive urban planning applications.
- Best-First Search (BestFS) emerged as a noteworthy algorithm in the study of network analysis for Pakistan's top 90 cities. Similar to A*, BestFS showcased impressive efficiency, particularly when accurate heuristic information was available. Leveraging heuristics effectively, BestFS demonstrated a commendable ability to significantly reduce the search space without compromising the accuracy of the shortest path, proving advantageous in navigating dense urban networks. The algorithm's heuristic-driven approach presents a valuable balance between exploration efficiency and path optimality. In urban planning applications, BestFS proved well-suited, especially when heuristic information was readily accessible. However, it introduces a trade-off between exploration efficiency and the strict guarantee of optimality, a consideration that becomes crucial in scenarios where finding the absolute shortest path is paramount.

V. Conclusion

This study underscores the importance of choosing the right algorithm based on the specific requirements and characteristics of the problem at hand. While each algorithm has its strengths, their effectiveness can vary greatly depending on factors such as graph size, density and the nature of the data. In the realm of urban planning, where the efficient movement within city networks is crucial, the selection of an appropriate algorithm can have significant practical implications.

Furthermore, our findings suggest a potential for hybrid approaches, where the strengths of different algorithms can be combined or applied in stages for more complex or large-scale applications. This approach could leverage the speed of A* for specific routes within a densely connected network, the thoroughness of Dijkstra's for comprehensive network analysis, and the insights from Best First's heuristic nature for future infrastructural development and planning.

Looking forward, this study opens avenues for further research, particularly in optimizing these algorithms for specific types of urban networks or integrating them with other AI techniques for more advanced applications. The potential for developing AI-driven tools to aid in urban planning and inter city path finding in rapidly growing urban centers like those in Pakistan is immense. Such tools could significantly contribute to smarter, more efficient city planning and management, ultimately enhancing the quality of life for urban residents.

References

[1] "Cities in Pakistan with Lat Long," www.latlong.net. https://www.latlong.net/category/cities-167-15.html (accessed Dec. 08, 2023).
[2] J. Louwers, "Calculate Geographic distances in Python with the Haversine method," Medium, Sep. 18, 2023. https://louwersj.medium.com/calculate-geographic-distances-in-python-with-the-haversine-method (accessed Dec. 11, 2023).
[3] N. Swift, "Easy A* (star) Pathfinding," Medium, Feb. 27, 2017. https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2
[4] A. Klochay, "Implementing Dijkstra's Algorithm in Python," Udacity, Oct. 12, 2021. https://www.udacity.com/blog/2021/10/implementing-dijkstras-algorithm-in-python.html
[5] "Best First Search (Informed Search)," GeeksforGeeks, May 03, 2017. https://www.geeksforgeeks.org/best-first-search-informed-search/