Enhanced Shortest Route Prediction Algorithm: A Hybrid Approach

Dr. S. Selvi[1] , Dhinoovika D[2] , Harshana R[3], Muthulakshmi R[4]

[1]Associate Professor, Department of Computer Science and Engineering, Government College of Engineering, Bargur,Krishnagiri, Tamil Nadu, India

[2,3,4] Pre-Final year students, + 5 Department of Computer Science and Engineering, Government College of Engineering, Bargur,Krishnagiri, Tamil Nadu, India

ABSTRACT

In this research, the shortest route prediction algorithm proposed a solution to find the simplest route with less time duration between two nodes. The proposed algorithm works only on the best route and concentrates on the nearest shortest node to determine the simplest path. The results show that the shortest route prediction approach significantly reduced the travel time. Efficiently determining the shortest path in a graph is a fundamental problem with numerous real-world applications. It explores the utilization of two prominent algorithms, A* and Dijkstra's, for solving the shortest path problem. A* combines the advantages of both uniform cost search and greedy best-first search, using heuristics to guide the search. On the other hand, Dijkstra's algorithm guarantees the shortest path by exploring all possible routes. In this study, we investigate the principles and implementation of these algorithms, highlighting their strengths and weaknesses. This research discusses how A* and Dijkstra's algorithms work for different pathfinding, and the results are analyzed.

Keywords: Shortest path, Fastest route, Minimum traveling, Route prediction algorithm.

INTRODUCTION

Enhanced Shortest Route Search Algorithm Using A* and Dijkstra's with Pygame efficiently finding the shortest route in a graph is a key challenge in the fields of computer science, gaming, and real-world applications such as logistics and transportation. In this introduction, it is proposed an innovative approach to solving this problem using the A* and Dijkstra's algorithms in combination with the Pygame library, providing both an effective and visually engaging solution. A* and Dijkstra's algorithms are two well-established methods for pathfinding. A* is known for its speed and accuracy, using heuristics to guide the search for the shortest path, while Dijkstra's algorithm guarantees the shortest route by exploring all possible paths. Combining these algorithms allows us to harness the strengths of both, resulting in an optimized route search.

The Pygame library, a popular choice for creating 2D games and visual simulations, is employed to enhance the user experience. By integrating Pygame, it is not only to find the shortest path efficiently but also visualize it dynamically and interactively. This brings a new level of engagement to route planning and makes it accessible in applications like games, educational tools, and GPS systems. In this project, it isdelved into the principles behind A* and Dijkstra's algorithms and demonstrates how to implement them in a Pygame environment. It explores how the Pygame library can be used to create interactive maps, display graphically appealing representations of routes, and provide real-time feedback to users.

The objective of this project is to provide a practical understanding of A* and Dijkstra's algorithms, offer hands-on experience with Pygame, and demonstrate the value of integrating these technologies for enhanced route searching. By the end of this project, readers will be equipped with the knowledge and tools to apply these techniques to a wide range of applications, from designing video game levels to optimizing logistical routes.

This paper is organized as follows. Section 2 presents the related work of the shortest route algorithm. Section 3 explains the proposed algorithm. Section 4 describes the workflow of the proposed algorithm. Performance analysis of the algorithm is presented in section 5. Results are concluded in Section 6.

LITERATURESURVEY

Different shortest-route algorithms implemented by different researchers are tabulated in Table 1. The advantages and disadvantages of their works are also presented.

Table 1. Collection of shortest route algorithms

SL.NO.	TITLE OF THE PAPER/AUTHOR/ YEAR	OBJECTIVE OF THE PAPER	PROPOSED ALGORITHM	PERFORMANCE METRICS	ADVANTAGES/ DISADVANTAGES
1	Path controlling of automated vehicles for system optimum on transportation with heterogeneous traffic systems. Zhibin chen-et.al. Emerging Technologies 2020.	Developed a path-control scheme to achieve the system optimum (SO) of the Network by controlling a portion of cooperative automated vehicles (CAVs) as per the routing principle.	Optimal- ratio control scheme. AVs are required to adopt the system optimization routing principle.	MCR (minimum control ratio) below 23%.	Finding the MCR of the scheme is delineated by a linear program, which can be solved by commercial solvers resulting in 96% only for autonomous driving vehicles.
2	A congestion Aware Tabu Search Heuristic to Solve the Shared Autonomous Vehicle Routing Problem. Prashanth Venkatraman & Michael W.Levin. Journal of Intelligent Transportation System 2021.	Solved the shared autonomous vehicle (SAV) routing problem under the effects of congestion in the road.	Developed a tabu search (TS) to heuristic solve for SAV routing problem.	The heuristic is found to produce encouraging results.	Using the swap procedure, travelers can switch paths where less traffic is found compared to other results.
3	Traffic congestion Aware Graph Based vehicle rerouting framework from Aerial Imagery. Erutugal Bayraktar- et.al. Engineering Applications of ArtificialIntelligence 2023	Proposed a modular rerouting for only one single vehicle framework composed of usual perception, property estimation, and trajectory optimization.	Employed Dijkstra's, A*, RRT, and RRT* to optimize cost.	RRT* achieves the fastest result by examining most of the possible options in less time.	They need to update the cost funding co-efficiently so that they can search the destination in the shortest time.
4	Survey of shortest Path Algorithms. Dr. Shaveta Bhatia- et.al. SSRG International Journal of Computer Science and Engineering.	The main objective to evaluate and compare different shortest-pathalgorithms.	Found the optional decision to investigate Dijkstra's algorithm and Floyd's algorithm.	Predicting and analyzing a simple path reduces the data pre-processing time and space cost.	It reduces time complexity as well as spacecomplexity. Sometimes it may not favor.
5	Efficient Shortest Path Index Maintenance on Dynamic Road Networks with Theoretical Guarantees. Dian Ouyang-et.al.	Computing the shortest path between two vertices is a fundamental problem in road networks that is applied in a wide variety of applications.	To achieve this goal, we proposed a shortcut-centric paradigm focusing on exploring a small number of shortcuts to maintain the shortest index. SS-Graph proposes a shortcut weight propagation mechanism.	The traffic level N-curve pattern from a convolutional neural network was reduced.	The shortest index for streaming update and batch update shows better efficiency. Sometimes it may not favor.

6	Stream Processingof Shortest PathQueryinDynamicRoadNetworks. Mengxuan Zhang et.al IEEE TransactionsonKnowledgeand DataEngineering2022	Solved thetrafficconditionwhichisstableoverashort periodand treated theissuedquerieswithin thatperiod as the stream ofquerysets.	Batch shortestpath algorithms have beenproposed toanswerasetofqueriestogetherusingshareablecomputation	The combinationofneuralnetworksprovides moreinformation ontraffic with betterresults.	A localcache thatimprovestheexistingglobalcachewitha highercache hitratio.

PROPOSED SYSTEM

This project aims to improve the accuracy of the heuristic shortest-path algorithm. Here, it implemented two algorithms namelyDijkstra's and A* Search Algorithm in Python to find the shortest route between two cells in a grid andvisualized their workflow using a 2D graphics module called Pygame. Twopopular algorithms are compared with the proposed algorithm called the enhanced version of the shortest route prediction algorithm and the time-dependent potentials algorithm are discussed in the background study. The workflow of the proposed system is shown in Figure 1. The proposed system issplit into several modules. They are explained hereunder:

1. Main Module (main() function)
Initializes Pygame and sets up the window for visualization. Manages user input for creating start, end, and obstacle nodes on a grid.Begins pathfinding algorithms (A* or Dijkstra) when the space key is pressed calculates and displays the execution time for updating neighbors and finding the shortest path. Resets the grid if the 'c' key is pressed.

2. Node Module (node.py)
Defines the Node class which represents each cell on the grid.Each node keeps track of its position, state (start, end, obstacle), and neighbors.

3. *A Module (a_star.py)
Contains the A* algorithm for finding the shortest path from the start to the end node. Utilizes a heuristic function (often Euclidean distance) to determine the optimal path.

4. Dijkstra's Module (dijkstra.py)
Contains Dijkstra'salgorithm for finding the shortest path from the start to the end node.Considers all possible paths and iterates through nodes to find the shortest path.

5. Drawing Functions (draw() and drawGridLines())
drawGridLines() draws grid lines on the Pygame window.draw() handles the visual representation of the grid and nodes.

6. Grid Builder (buildGrid())
Generates a grid structure based on the provided row and width parameters.Initializes the grid with node instances representing each cell.

7. Helper Functions
getClickedPosition(): Converts the mouse click position to a grid cell. cg(): Counts the number of cells marked as a certain color (in this case, purple) in the Pygame window.

Each module or function serves a specific purpose within the path-finding visualizer, from creating the grid and nodes to managing user interactions and executing path-finding algorithms.

WORKFLOW OF THE PROPOSED SYSTEM

This workflow of the shortest route prediction algorithm involves incorporating elements from A* and Dijkstra's algorithms to optimize the search for the shortest path in a graph.The workflow diagram captures the dynamic nature of the algorithm, where it can adapt its strategy based on specific conditions, using A* when the heuristic is beneficial and switching to Dijkstra's when it's not. The decision points and loops in the workflow represent the iterative nature of the search process.

A* And Dijkstra's Algorithm
A*algorithm is a widely used heuristic search algorithm that aims to find the shortest path from a start node to an end node on a graph. It combines the advantages of both uniform cost search and greedy best-first search. It employs a

heuristic function to estimate the cost of reaching the goal from a specific node. In this project, A* is implemented to find the shortest path on the map of Dijkstra's Algorithm.

Dijkstra's algorithm is a classical algorithm for finding the shortest path in a graph by exploring all possible routes systematically. It guarantees the shortest path by maintaining a priority queue of nodes with the shortest known distance. Dijkstra's algorithm is implemented to compare its performance with A*.

EXPERIMENTAL EVALUATION

The minimum and recommended criteria for hardware requirement can run the system but low system performance Processor (CPU) 1 GHz or faster CPU, Memory (RAM) 4 GB RAM, Hard drive free space 16 GB. This project code is a Python implementation of a pathfinding visualizer using Pygame.

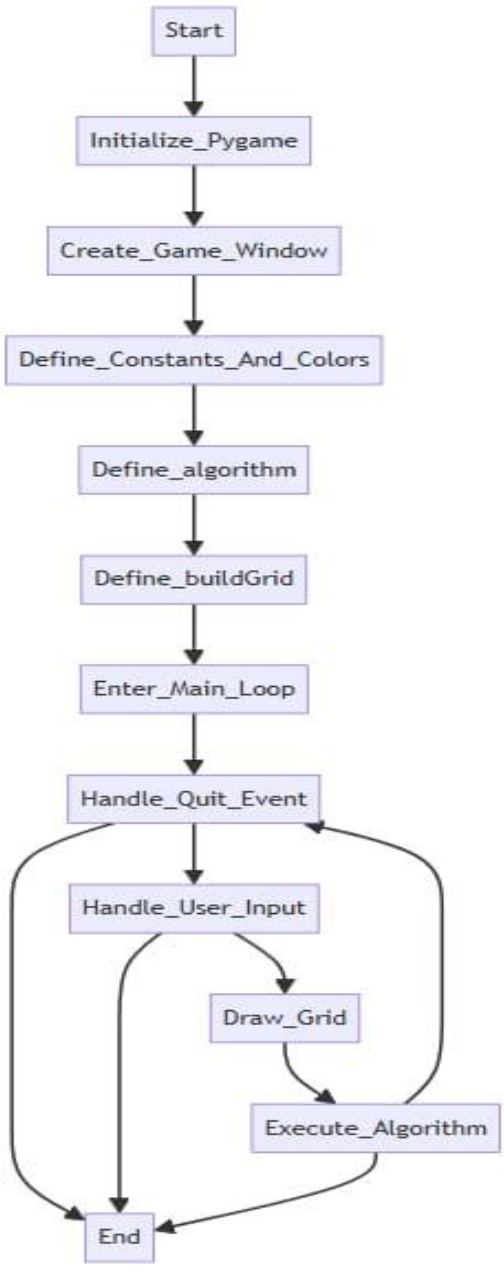


Figure 1. The Workflow of the Proposed System

PYGAME INTEGRATION

Pygame is a Python library designed for creating 2D games and interactive multimedia applications. In this project, Pygame is utilized to create a graphical user interface for visualizing the route search process. The Pygame library is used to display the map or grid on which the route search takes place. Users can interact with this map by specifying the start and end points for the route. Pygame is a popular Python library designed for creating 2D games and interactive multimedia applications. It provides a range of functions and tools for game development and multimedia programming. Pygame is cross-platform and works on various operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of developers.

Pygame offers comprehensive support for 2D graphics, including drawing shapes, images, and text, as well as handling image loading and manipulation. It allows developers to add sound effects and music to their games or applications. You can play and control audio files with ease. Pygame simplifies user input handling, including keyboard and mouse events. This is essential for creating interactive applications. It includes basic physics and collision detection functionality, making it suitable for game development. It helps manage object interactions and game physics. Pygame supports sprite handling and animation, making it easier to create animated characters and objects within games. You can create game windows, manage screen resolution, and handle full-screen or windowed modes. Pygame utilizes an event-driven model, where events like user input, timers, and system events are processed in a loop.

There's a wide range of community-contributed libraries and extensions available for Pygame, expanding its capabilities for specific purposes. Pygame has a strong community and extensive documentation, making it a user-friendly choice for developers. There are many tutorials, forums, and resources available to help users get started. Pygame is open-source and released under the LGPL (GNU Lesser General Public License), which means it can be used for both personal and commercial projects without significant licensing constraints. It is often used for educational purposes, game jams, rapid prototyping, and even the development of full-fledged 2D games. Its simplicity and versatility make it a valuable tool for those interested in 2D game development and multimedia programming using Python.

2D GRID

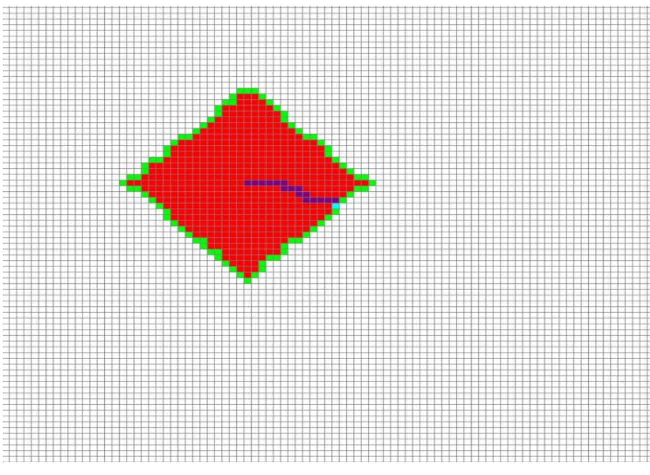
A 2D grid, or two-dimensional grid, is a data structure used to represent information in a two- dimensional space. It is essentially an array or matrix with rows and columns, where each cell in the grid can store data or values. 2D grids are commonly used in various fields, including computer graphics, game development, data visualization, and many other applications.

A 2D grid consists of rows and columns, forming a grid of cells. Each cell can be identified by its row and column indices Cells in a 2D grid can be addressed using a pair of coordinates, typically (x, y), where 'x' represents the column index and 'y' represents the row index. The origin (0, 0) is often at the top-left corner. Each cell in the grid can store data, such as numbers, characters, objects, or any other relevant information. In most cases, 2D grids have a rectangular shape, meaning that all rows have the same number of cells, and all columns have the same number of cells. However, it is possible to have irregular grids where the number of cells in each row or column varies.

PERFORMANCE ANALYSIS

This project code is a Python implementation of a pathfinding visualizer using Pygame. The results are shown inFigures 2-9. The pathfinding results using the Dijikstra's, A*, and Enhanced version of the algorithm without the obstacles and with the obstacles are displayed. Also,several gridswere encountered during the pathfinding and the time taken for finding the different shortest route algorithms is calculated. It is observed that the enhanced version proves that it simplifies the pathfinding and finds the shortest route which is shorter than comparing the other existing popular algorithms. The values obtained by using different algorithms are tabulated in Table 2. The performance of these algorithms is shown graphically in Figure 10.

SHORTEST PATH WITHOUT OBSTACLES



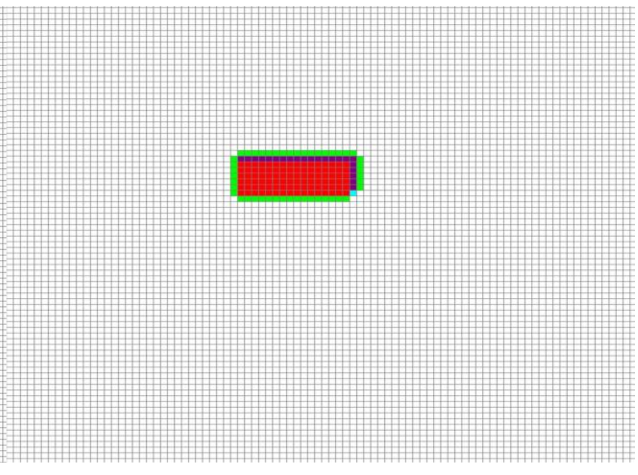


Figure 2. Dijkstra's algorithm **Figure 3. A* algorithm**

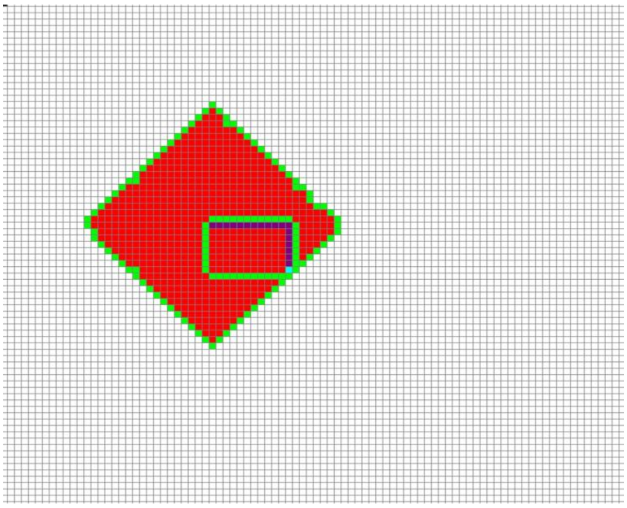


Figure4. Enhanced Dijkstra and A* algorithm

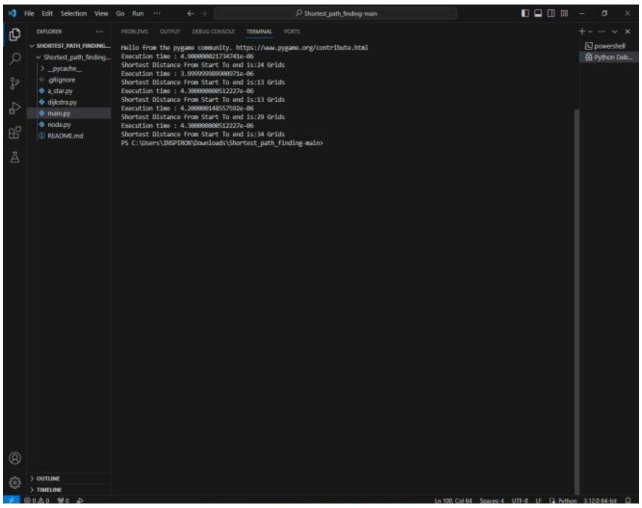


Figure5. Grid count and time execution

SHORTEST PATH WITH OBSTACLE

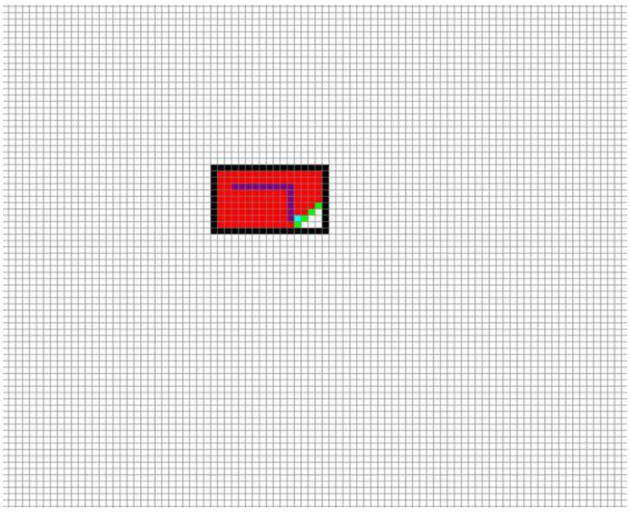


Figure. 6 Dijkstra's algorithm

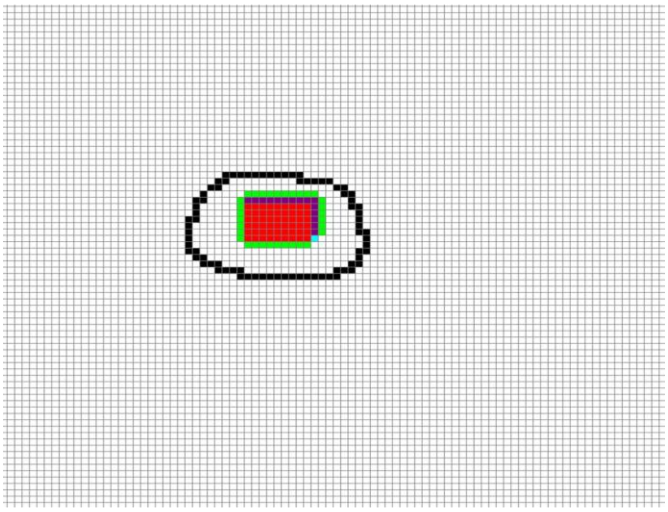


Figure. 7 A* algorithm

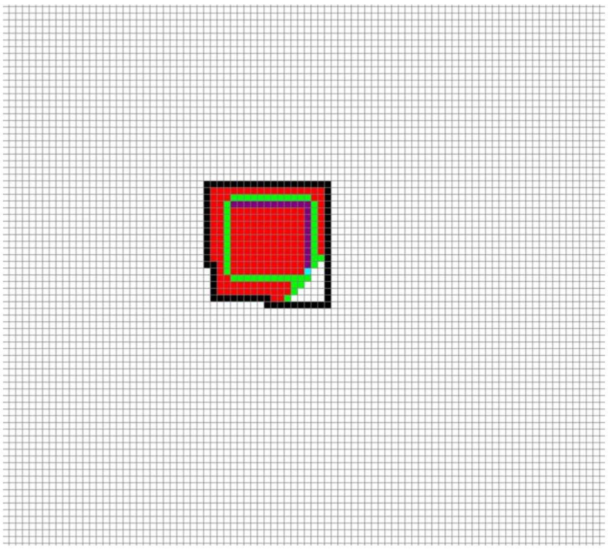


Figure 8. Enhanced Dijkstra and A* algorithm

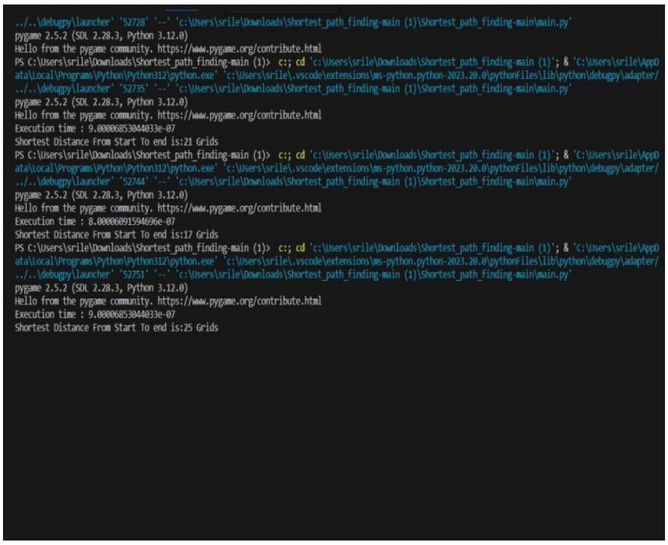


Figure 9. Grid count and time execution

Table 2. Comparison of Dijkstra's, A * and shortest prediction Aalgorithms

S.No.	Grid Count / Algorithm	Dijkstra's (ms.)	A-Star (ms.)	Shortest prediction (ms.)
1.	10	2000	1000	100
2.	25	3000	2500	500
3.	50	5000	3000	1500
4.	100	7500	6000	2500

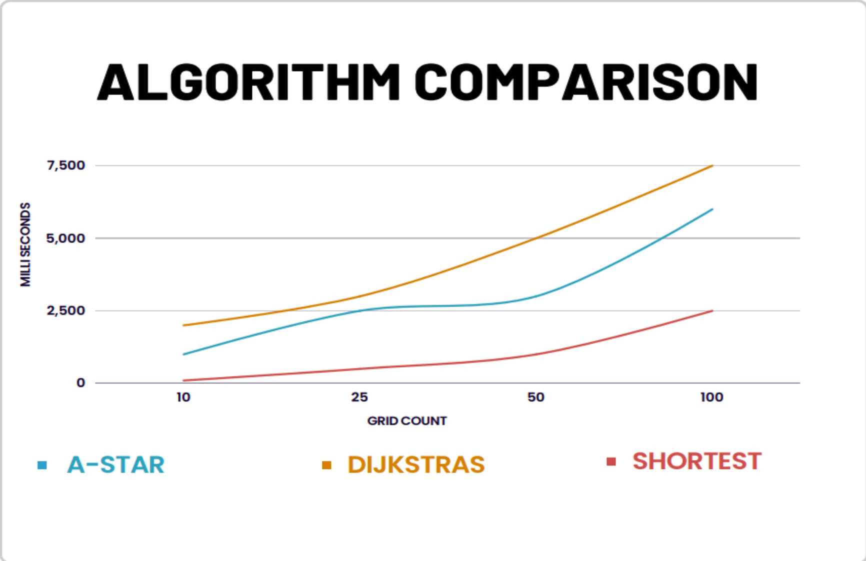


Figure 10. Performance Analysis Graph

CONCLUSION

The Enhanced version of the shortest route A*search is an effective method to find the shortest path by pre-processing the previous and current traffic data with various input types and identifying the shortest path through graph theory. The starting node concentrates on finding the nearest targeted goal and then processes each step. The main advantage of the proposed enhanced version of the shortest route A* search algorithm is that it does not waste time on spending any other unwanted nodes. Compared with other algorithms such as Time-dependent A*potentials, Enhanced Dijkstra's algorithm, and Heap-based/Enhanced Bellman-Ford algorithm, results in less accuracy displayed in graphical representation. The enhanced version of the shortest route A*search makes the fastest prediction of the shortest route with 95% of high accuracy.

REFERENCES

[1]. Zhibin Chen et al., "Path Controlling of Automated Vehicles for System Optimum on Transportation Networks with Heterogeneous Traffic Streams," Transportation Research Part C: Emerging Technologies, vol. 110, pp.312-329, 2020.

[2]. Prashanth Venkatraman, and Michael W. Levin, "A Congestion-Aware Tabu Search Heuristic to Solve the Shared Autonomous Vehicle Routing Problem," Journal of Intelligent Transportation Systems, vol. 25, no. 4, pp. 343-355, 2021.

[3]. Ertugrul Bayraktar et al., "Traffic Congestion-Aware Graph-Based Vehicle Rerouting Framework from Aerial Imagery," Engineering Applications of Artificial Intelligence, vol. 119, p.105769.

[4]. Dr. Shaveta Bhatia, "Survey of Shortest Path Algorithms," SSRG International Journal of Computer Science and Engineering, vol.6, no. 11, pp. 33-39, 2019.

[5]. Dian Ouyang et al., "Efficient Shortest Path Index Maintenance on Dynamic Road Networks with Theoretical Guarantees," Proceedings of the VLDB Endowment, vol. 13, no. 5, pp. 602-615, 2020.

[6]. Mengxuan Zhang et al., "Stream Processing of Shortest Path Query in Dynamic Road Networks," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 5, pp. 2458-2471, 2022.