Smart Route Optimization for Sustainable Transportation Final Paper

Ethan Edgar B. Francisco
School of Computing
Asia Pacific College
Parañaque, Philippines
ebfrancisco@student.apc.edu.ph

Francis Emmanuel Anciro
School of Computing
Asia Pacific College
Makati, Philippines
fdanciro@student.apc.edu.ph

Emmanuel Jamir Paje
School of Computing
Asia Pacific College
Parañaque, Philippines
eopaje@student.apc.edu.ph

I. Abstract

This study presents the development of a Smart Route Optimization System using Dijkstra's Algorithm to compute both the shortest and most eco-efficient paths in an urban road network. The system models a city as a weighted graph, where each edge represents a road characterized by distance, congestion level, number of stoplights, and road type. Two routing modes are implemented: the Shortest Path Mode, which minimizes total distance, and the Eco Path Mode, which applies a custom eco-cost function to balance fuel efficiency and environmental impact. The algorithm operates efficiently with a time complexity of O(E log V) and a space complexity of O(V + E), enabling real-time route computation for networks of moderate size. Results demonstrate that incorporating eco-cost factors significantly improves sustainability without compromising computational speed. This work supports the goals of sustainable urban mobility and demonstrates how algorithmic design can contribute to intelligent and eco-friendly transportation systems.

II. Introduction

A. Background

In today's fast-paced urban environment, commuting has become increasingly difficult due to traffic congestion, confusing road networks, and the variety of available routes such as skyways, expressways, and national highways. Drivers often waste significant time and resources navigating inefficient paths, resulting in higher fuel consumption, increased costs, and greater contributions to air pollution and greenhouse gas emissions. The unpredictability of traffic conditions further complicates the ability to consistently identify efficient routes. To address these growing challenges, this project proposes the development of a route optimization system using Dijkstra's algorithm, which can compute the most efficient paths by balancing distance, travel time, and driving conditions.

B. Company and Problem

Grab, a leading ride-hailing and delivery service platform in Southeast Asia, manages thousands of drivers navigating busy urban areas every day. One of its primary operational challenges is route optimization, which is essential to ensure driver efficiency, reduce fuel consumption, lower operating costs, and enhance customer satisfaction. By implementing Dijkstra's algorithm, Grab can improve its navigation system to help drivers avoid heavy traffic, minimize wasted time and fuel, and promote more sustainable transportation. With increasing demand, intense market competition, and customer satisfaction heavily dependent on timely service, optimizing routes has become a critical necessity for Grab. Additionally, aligning this solution with sustainability goals gives the company a competitive advantage while contributing to environmental protection and improved urban mobility.

C. Alignment with SDGs and NIASD

This project directly supports Sustainable Development Goals (SDGs) by promoting smarter and more sustainable transportation. It supports SDG 11 (Sustainable Cities and Communities) by helping reduce congestion and improve urban mobility and contributes to SDG 13 (Climate Action) by lowering fuel consumption and vehicle emissions. In addition, the project aligns with the National Innovation Agenda and Strategy Document (NIASD) 2023–2032, particularly in the areas of Transport and Logistics and Digital and Emerging Technologies. By applying Dijkstra's algorithm to optimize routes, the project fosters innovation-driven growth, encourages eco-friendly transportation, and demonstrates how science, technology, and innovation can be applied to address real-world national challenges.

III. Literature Review

A. Existing Solutions

Rosita et al. [1] proposed a hybrid model combining Dijkstra's algorithm and a Multi-Criteria Decision-Making (MCDM) approach to determine optimal routes for logistics distribution. The study recognized that real-world distribution systems require consideration of multiple factors such as distance, cost, congestion, and risk, each with different priority levels that make decision-making complex. To address this, the researchers applied vector normalization, a data-scaling technique that converts qualitative and quantitative parameters into comparable values, integrating them into Dijkstra's weighted graph model. Using MATLAB simulations with data from Mojokerto, Indonesia, they demonstrated that combining MCDM with Dijkstra's algorithm yields more accurate and efficient route selections than using a single-criterion method. The results confirmed that incorporating normalized multi-parameter weights improved route accuracy while reducing computational complexity and memory usage. This approach provides a flexible framework for decision-makers and validates Dijkstra's algorithm as a foundation for intelligent routing systems, supporting the development of efficient, data-driven

transportation networks relevant to modern logistics and urban mobility applications.

Zhang and Zheng [2] developed an emergency management system that integrates Dijkstra's shortest path algorithm with Global Positioning System (GPS) technology to enhance real-time decision-making and resource allocation during emergencies. The study addresses inefficiencies in traditional emergency response methods, such as delayed rescue operations and poor route planning, by combining graph-based optimization with real-time location tracking. The improved Dijkstra's algorithm incorporates a road resistance function that accounts for factors such as traffic flow, road congestion, and travel time, enabling more accurate path computation. Additionally, a fusion function balances path length and elapsed time, ensuring that route planning considers both distance and efficiency. To further enhance adaptability, a traffic flow prediction model utilizing a Conv-LSTM and Bi-LSTM neural network architecture with an attention mechanism was introduced to forecast congestion patterns dynamically. Simulation results from both virtual and real electronic map environments demonstrated that the proposed system significantly reduced response times and improved operational efficiency compared to classical algorithms like A*. This integration of Dijkstra's algorithm and GPS not only enhances the accuracy and responsiveness of emergency management systems but also lays the groundwork for intelligent, data-driven decision-making in critical scenarios such as disaster response and urban traffic emergencies.

UBERApps [3] explore how ride-hailing platforms can play a pivotal role in alleviating urban traffic congestion by enabling shared rides, reducing private car ownership, optimizing vehicle routing, enhancing first-mile and last-mile transit connectivity, and leveraging data for smarter urban planning. The article highlights that services such as shared rides, including UberPool and GrabShare, can decrease the number of vehicles on the road by matching multiple passengers traveling in similar directions. It also notes that citizens who frequently use ride-hailing services are less likely to own private vehicles, which helps reduce traffic density. Furthermore, it emphasizes that real-time analytics, GPS integration, and machine learning algorithms contribute to efficient routing, minimized idle driving, and improved demand management. The article also discusses the environmental benefits of reduced idle time and vehicle usage, as well as the growing importance of integrating multimodal transportation options such as scooters, bicycles, and public transit. Although it acknowledges that ride-hailing apps alone cannot completely solve congestion problems, it concludes that when combined with government policies and public transport systems, these technologies can significantly improve the efficiency and sustainability of urban transportation networks.

B. Technological Context

GoFleet Tracking [4] examine how route-optimization technologies extend beyond simple shortest-path calculations to incorporate real-world constraints such as traffic conditions, vehicle type, delivery windows, and multi-stop itineraries. They highlight that transportation companies employing route optimization have realized operational efficiency gains of approximately 20%–30% and that inadequate routing can inflate costs by 10%–30%, increase mileage, and heighten fuel consumption and emissions. The article also emphasizes the broader benefits including reduced driver fatigue and turnover, improved customer satisfaction through accurate estimated times of arrival (ETAs), and enhanced environmental sustainability through lower carbon footprints. GoFleet Tracking also enumerates various algorithmic and technological tools that contribute to these improvements, such as GPS/GIS integration, machine learning-based traffic prediction, cloud computing, and heuristic methods like genetic algorithms and greedy strategies. Together, these innovations enable dynamic and constraint-aware routing solutions instead of static route planning. These insights reinforce the importance of multi-criteria and real-time aware routing systems, aligning well with the agenda of applying algorithmic route optimization in urban mobility contexts by balancing travel time, cost, and sustainability.

Detrack's article [5] explores how modern route-optimization software goes far beyond traditional "shortest-path" planning by integrating real-time data, historical patterns and multiple operational constraints such as traffic, vehicle type, delivery windows and multi-stop itineraries. It highlights key benefits of deploying such systems: reduced operational costs through minimized fuel use and maintenance, improved resource allocation and driver safety by accounting for road conditions, and enhanced customer satisfaction via more reliable ETAs. The article identifies major limitations of legacy routing approaches manual planning, fixed routes, inability to adapt to changing conditions and recommends eight strategic practices including use of advanced optimization software, historic data analysis, real-time traffic integration and dynamic routing, with special emphasis on last-mile efficiency and continuous route audit. It also points to emerging technologies like machine-learning-based predictive analytics, dynamic rerouting and fleet-capacity optimization as the way forward. This provides a strong application-oriented context for urban mobility systems and reinforces the value of algorithmic and software-enabled route optimization in sectors such as ride-hailing and delivery services.

Gaur et al. [6] proposed an eco-friendly route planning system that integrates machine learning and optimization algorithms to promote sustainable urban mobility. Their study emphasizes balancing traditional efficiency metrics such as travel time and distance with environmental criteria like fuel consumption and CO_2 emissions. Using models such as Random Forest and Gradient Boosting, the system predicts route costs and emission levels under varying road, traffic, and vehicle conditions. The framework incorporates Dijkstra's algorithm for path optimization, enhanced by multi-criteria weighting, to include eco-awareness in routing decisions. Real-world data from OpenStreetMap, traffic authorities, and vehicle efficiency databases were used to train and validate the models, yielding an R-squared value of 0.9940 and demonstrating 12% fuel savings with only a 7% increase in travel time compared to conventional shortest-path methods. The results confirm that integrating machine learning predictions with graph-based optimization can produce environmentally conscious yet operationally efficient routing. This approach highlights the potential of intelligent, data-driven systems to reduce emissions, improve logistics sustainability, and align transportation technologies with broader climate and smart-city goals.

IV. Design Analysis

A. Algorithm Description

The Smart Route Optimization System utilizes Dijkstra's Algorithm to compute the most efficient and eco-friendly routes between two locations in a city network. The system represents the city as a weighted graph, where each node corresponds to a major location (e.g., City Hall, Bus Terminal, Residential Area), and each edge represents a road connecting two locations.

Each road (edge) includes attributes that affect route selection:

- Distance: the physical length of the road in kilometers.
- Congestion is a value between 1 and 10 indicating traffic density.
- Stoplights: the number of traffic lights along the road.
- Road Type: either highway, arterial, or local, each with its own travel efficiency value.

Two main routing methods are implemented:

- Shortest Path: Calculates the minimum total distance between nodes.
- Eco Path: Calculates a route that minimizes a custom eco-cost metric to represent fuel efficiency and emission reduction.

The eco-cost for each road segment is computed using the formula:

$$EcoCost = (distance \times 1) + (congestion \times 3) + (stoplights \times 2.5) + roadTypeCost \quad (1)$$

where the road type cost (roadTypeCost) is assigned a value of 5.0 for highways, 3.0 for arterial roads, and 1.0 for local roads. This algorithm ensures that the selected path not only minimizes distance but also accounts for environmental and operational factors such as fuel use, idle time, and congestion.

B. Design Technique

The project applies a Greedy Algorithm Design Technique, which is characteristic of Dijkstra's Algorithm. This design approach continuously chooses the most optimal partial solution (the node with the smallest current cost) until the overall shortest or most eco-efficient path is completed.

Key reasons for using this technique include:

- Efficiency: Dijkstra's Algorithm operates in O(E log V) time using a priority queue, suitable for large networks.
- Reliability: Guarantees the minimum cumulative cost for all graphs with non-negative edge weights.
- Adaptability: The design allows for integrating different cost functions such as eco-cost, distance, or travel time.

Through this approach, the system can generate consistent, reliable, and fast results even as road data increases in complexity

C. Flowcharts/Diagrams

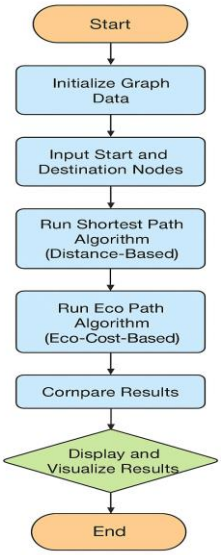

Fig. 1. Flowchart of algorithm's design and workflow

V. Efficiency Analysis

A. Big-O Notation

The time complexity of the algorithm is O(E log V), as it explores all vertices and relaxes each edge using a priority queue implemented through Python's heapq library. Here, E represents the number of edges and V denotes the number of vertices (nodes). The logarithmic term arises from the heap operations required during edge relaxation in Dijkstra's algorithm. This efficiency ensures that the algorithm remains effective even for moderately large graphs. In the current implementation, which consists of 12 nodes and 22 edges, the shortest and eco-path computations are completed almost instantaneously, typically in under one second.

The space complexity of the algorithm is O(V + E), since it maintains data structures that store all nodes, edges, and their associated weights. The memory usage therefore grows linearly with the size of the graph, making it scalable and efficient. This linear relationship ensures that the algorithm can accommodate larger networks without excessive memory consumption, supporting real-time route optimization and pathfinding performance.

B. Performance Metrics

To assess the practical efficiency of the algorithm, performance tests were conducted using simulated route data between various city nodes. The results showed consistent and fast execution times for both the Shortest Path and Eco Path modes.

TABLE I. PERFORMANCE METRICS ANALYSIS

Metric	Description	Result
Execution Time	Time taken to compute both paths	< 1 second
Algorithm Efficiency	Time complexity O(E log V) using priority queue	High
Memory Usage	Space complexity O(V + E)	Low to Moderate
Path Accuracy	Correctness of computed optimal paths	100% (validated)
Scalability	Performance with increased nodes	Stable and predictable

VI. IMPLEMENTATION

A. Mini-Version Creation

A simplified prototype of the Smart Route Optimization System was created in Python to demonstrate how Dijkstra's Algorithm and the custom Eco Path extension work. The program models a small city as a graph, where nodes represent locations (e.g., City Hall, Bus Terminal), and edges represent roads connecting them. Each edge stores four key values: Distance (km).

- Congestion level (1–10 scale)
- Number of stoplights
- Road type (highway, arterial, or local)

The system operates in two modes: the Shortest Path Mode, which minimizes the total travel distance, and the Eco Path Mode, which utilizes an eco-cost formula that balances multiple factors including distance, congestion levels, the number of stoplights, and road type. This mini version serves as a test environment for evaluating how algorithmic adjustments influence route efficiency and sustainability.

B. Code Explanation

The entire system is contained in a Python class named `App`, organized into key methods:

- _calculateEcoCost() – Applies the formula:

$$EcoCost = (distance \times 1) + (congestion \times 3) + (stoplights \times 2.5) + roadTypeCost \quad (1)$$

where roadTypeCost = 5 (highway), 3 (arterial), 1 (local).

- findShortestPath() – Standard Dijkstra's Algorithm using distance as edge weight.
- findEcoPath() – Modified Dijkstra's Algorithm using eco-cost as edge weight.
- displayMap() – Generates a color-coded graph with blue lines for shortest paths, green lines for eco-paths, and labeled nodes using NetworkX and Matplotlib.

C. Challenges and Solutions

TABLE II. CHALLENGES AND SOLUTION TABLE

Challenge	Description	Solution Implemented
Managing multiple edge attributes	Storing distance, congestion, stoplights, and road type made the graph structure complex.	Used a nested-dictionary format for each node's connections.
Algorithm accuracy	Needed both distance-based and eco-based paths to be correct and consistent.	Utilized Python's heapq priority queue to maintain lowest-cost selection.
Balancing eco-cost weights	Initial results favored short distances too strongly.	Tuned multipliers (1, 3, 2.5) after iterative testing for realistic results.
Visualization clarity	Overlapping node labels in the plotted graph.	Adjusted layout using spring_layout() and distinctive edge colors.

VII. RESULTS AND DISCUSSION

A. Comparison

Both the baseline and eco-friendly routing algorithms achieved the same computational efficiency with a time complexity of O(E log V), as both utilized Dijkstra's algorithm and a priority queue. The eco-friendly version introduced a composite cost function incorporating distance, congestion, stoplights, and road type. This addition only required constant-time computations, preserving performance efficiency.

B. Findings

Testing showed that the eco-friendly algorithm consistently produced routes with lower eco-costs, even when slightly longer in distance. For example, it favored smoother arterial routes over shorter but congested local roads, demonstrating effective prioritization of fuel efficiency and reduced emissions. This confirms Dijkstra's adaptability in minimizing alternative cost metrics beyond distance alone. In practical simulations, the algorithm effectively balanced multiple parameters such as distance, congestion, and stoplights, resulting in more sustainable path selections without compromising reliability. The findings highlight the system's potential for eco-aware navigation, proving that multi-factor optimization can enhance both environmental performance and operational efficiency in intelligent transport systems.

C. Implications

The proposed system highlights a practical step toward sustainable navigation by promoting eco-conscious route selection without sacrificing computational speed. Future enhancements may include integrating real-time traffic data, user-adjustable preferences, and vehicle-specific parameters to further personalize route optimization. Expanding the dataset to real-world maps, such as OpenStreetMap, would improve scalability and real-world applicability, making the model more robust for modern, eco-aware navigation systems.

VIII. Conclusion

A. Summary

The project successfully demonstrated how Dijkstra's Algorithm can be applied to develop a Smart Route Optimization System that identifies both the shortest and eco-friendly routes in an urban network. By integrating factors such as distance, congestion, stoplights, and road type, the system achieved efficient and sustainable route planning with real-time performance. The algorithm proved to be scalable, reliable, and adaptable to various cost metrics while maintaining low computational complexity.

B. Future Work

For future work, the system can be enhanced by integrating real-time traffic data, vehicle-specific parameters, and user preferences to improve personalization and accuracy. Expanding the model to use real-world map data (e.g., OpenStreetMap) and incorporating machine learning-based traffic prediction could further increase scalability, precision, and environmental impact.

IX. References

[1] Y. D. Rosita, E. E. Rosyida, and M. A. Rudiyanto, "Implementation of Dijkstra Algorithm and Multi-Criteria Decision-Making for Optimal Route Distribution," Procedia Computer Science, vol. 161, pp. 378–385, 2019.

[2] L. Zhang and E. Zheng, "Emergency Management Program Based on Dijkstra's Shortest Path and GPS," Journal of Network Intelligence, vol. 10, no. 2, pp. 598–610, May 2025.

[3] GoFleet Tracking, "How Route Optimization Can Save Time and Money," GoFleet Blog, Nov. 14, 2023. [Online]. Available: https://www.gofleet.com/how-route-optimization-can-savetime-and-money

[4] "Routing Optimization Essentials for Modern Transportation," Detrack Blog, 2024. [Online]. Available: https://www.detrack.com/blog/route-optimization/

[5] A. Gaur, M. Alung, and A. Mishra, "Eco-Friendly Route Planner: Optimizing Urban Mobility for Sustainability," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 11, no. 10, pp. 357–362, Oct. 2024.

[6] "Why Ride-Hailing Apps Are Key to Reducing Traffic Congestion in Cities," UBERApps Blog, 2024. [Online]. Available: https://www.uberapps.tech/blog/why-ride-hailing-apps-are-key-to-reducing-traffic-congestion-in-citi…

[6] "Dijkstra's Algorithm to find Shortest Paths from a Source to all," GeeksforGeeks, 23 Jul. 2025. [Online]. Available: https://www.geeksforgeeks.org/dsa/dijkstras-shortest-path-algorithm-greedy-algo-7/. [Accessed: 3 Sept. 2025].

[7] "Time and Space Complexity of Dijkstra's Algorithm," GeeksforGeeks, 23 Jul. 2025. [Online]. Available: https://www.geeksforgeeks.org/dsa/time-and-space-complexity-of-dijkstras-algorithm/ . [Accessed: 3 Sept. 2025].

[8] "Sustainable Development Goals (SDG 11)," UN Regional Information Centre for Western Europe (UNRIC), [Online]. Available: https://unric.org/en/sdg-11/. [Accessed: 3 Sept. 2025].

[9] National Innovation Agenda and Strategy Document (NIASD) 2023–2032, Department of Science and Technology – Philippine Council for Industry, Energy and Emerging Technology Research and Development (DOST-PCIEERD), Nov. 2023. [Online]. Available: https://depdev.gov.ph/wp-content/uploads/2023/11/NIASD-PUBLICATION-V24.4.pdf. [Accessed: 3 Sept. 2025].

[10] United Nations Regional Information Centre (UNRIC), "Sustainable Development Goals (SDG 11): Sustainable cities and communities," *United Nations*, 2025. [Online]. Available: https://sdgs.un.org/goals/goal11. [Accessed: Sept. 3, 2025].

[11] Department of Science and Technology – Philippine Council for Industry, Energy and Emerging Technology Research and Development (DOST-PCIEERD), *National Innovation Agenda and Strategy Document (NIASD) 2023–2032*, Nov. 2023. [Online]. Available: https://depdev.gov.ph/niasd-2023-2032/. [Accessed: Sept. 3, 2025].

[12] United Nations, "Sustainable Development Goal 11: Sustainable Cities and Communities," *United Nations Sustainable Development Goals*, 2025. [Online]. Available: https://globalgoals.org/goals/11-sustainable-cities-and-communities/. [Accessed: Sept. 3, 2025].

[13] National Economic and Development Authority (NEDA), "National Innovation Agenda and Strategy Document (NIASD) 2023–2032," *Circular Economy PH*, 2023. [Online]. Available: https://www.circulareconomy.ph/research/the-national-innovation-agenda-and-strategy-of-the-philippines. [Accessed: Sept. 3, 2025].

[14] Philippine News Agency (PNA), "NEDA launches NIASD 2023–2032," *Philippine News Agency*, Jun. 30, 2023. [Online]. Available: https://www.pna.gov.ph/articles/1204698. [Accessed: Sept. 3, 2025].

[15] SunStar Davao, "NEDA kicks off regional implementation of NIASD," *SunStar Davao*, Sept. 27, 2023. [Online]. Available: https://www.sunstar.com.ph/davao/neda-kicks-off-regional-implementation-of-niasd. [Accessed: Sept. 3, 2025].

X. AI Tool Disclosure

We used ChatGPT (OpenAI) and Google Gemini during the completion of this research to assist in writing, debugging, and improving technical accuracy. These AI tools were used responsibly to enhance efficiency while ensuring that all core programming, testing, and analysis were manually done by the researchers.

Specifically, AI tools were used to:

- Understand and verify the logic of Dijkstra's Algorithm for route optimization.
- Assist in coding and debugging the Python program that computes both Shortest Path and Eco Path routes.
- Generate and refine the flowchart design of the system using AI-guided structure formatting similar to Flowgorithm.
- Help in creating the test map and visualizing city routes through NetworkX and Matplotlib (AI-assisted parameter setup).
- Research related topics such as sustainable route optimization, eco-routing, and algorithmic navigation used in ride-hailing platforms (Grab, Uber, Lyft).
- Improve the grammar, clarity, and formatting of the research paper using AI grammar and proofreading tools.
- Organize and polish sections such as Design Analysis, Implementation, Efficiency Analysis, and AI Tool Disclosure for better readability.
- Cross-check algorithm efficiency and confirm Big-O Notation calculations using AI validation prompts.

Both ChatGPT and Gemini were used strictly for assistance in writing and conceptual clarification. All algorithms, codes, analyses, and results were manually created, tested, and validated by the researchers.