Route Optimization for Municipal Solid Waste Collection in Kuppiyawatha East, Sri Lanka, Using Dijkstra's Algorithm

Sanudi Manjusri

Sri Lanka Institute of Information Technology

hs21340482@my.sliit.lk

Kavinya De Silva

Sri Lanka Institute of Information Technology

hs21939396@my.sliit.lk

Suban Kokilakumar

Sri Lanka Institute of Information Technology

hs21938788@my.sliit.lk

Abstract

Efficient waste collection and transportation are critical for urban solid waste management, yet many municipalities in Sri Lanka, including Colombo City, struggle with inefficient routing, increased operational costs, and environmental impacts. This study focuses on optimizing waste collection routes in Kuppiyawatha East, Colombo District, Sri Lanka, using Dijkstra's Algorithm to minimize the total distance travelled while ensuring complete coverage of all waste collection points. A distance matrix was constructed based on real-world locations, and the algorithm iteratively processed nodes to determine the most efficient path. The optimized route covered all required nodes with a total travel distance of 3,450 meters, demonstrating a significant improvement over unstructured routing method. The findings highlight the effectiveness of mathematical modelling in waste transportation optimization, with implications for reducing fuel consumption, labour costs, and environmental emissions. This research provides a data-driven approach to improving waste collection efficiency in Colombo City, offering a foundation for sustainable urban waste management practices in Sri Lanka.

Keywords: Climate action, Dijkstra Algorithm, Operational Efficiency, Route Optimization, Waste Management.

1. Introduction

Solid waste management is a growing global challenge, with the increasing volume and complexity of waste posing significant risks to ecosystems and human health. An estimated 11.2 billion tonnes of solid waste is collected worldwide annually, with poor management contributing to air pollution, water contamination, and greenhouse gas emissions, particularly methane and carbon dioxide. In developing countries, inefficient collection and disposal methods further exacerbate environmental and health concerns (UN enviroment Program, n.d.).

Sri Lanka generates approximately 7,000 metric tonnes of solid waste daily, with the Western Province accounting for nearly 60% of total waste production. Each individual produces an average of 0.4 to 1 kg of waste per day, yet only 50% of the waste is collected, according to the Waste Management Authority and the Central Environmental Authority. Over the last two decades, the Sri Lankan government has attempted various waste management strategies, including sanitary landfills and waste-to-energy projects. However, the lack of a unified and coherent strategy has led to inconsistent policies and inefficient waste collection systems, resulting in severe environmental degradation and public health crises. Tragic incidents, such as the 2017 Meethotamulla garbage dump collapse, which killed 30 people and destroyed over 100 homes, highlight the urgency of improving waste management systems (EFL Admin, 2018). One of the most pressing issues in waste management is the inefficiency in waste collection and transportation. In Sri Lanka, waste collection relies primarily on door-to-door collection, communal storage bins, kerbside collection, and block collection systems (Basnayake & Visvanathan, 2013).

Moreover, seven waste management steps were identified to streamline the management and handling of MSW in the Western Province. They are namely; the management of waste at source, proper collection/acceptance of waste from the generating point, cleaning of streets and public places, providing of adequate infrastructure facilities, improved system of waste transportation, use of collected waste as a ‘‘resource’’ and the providing of proper final disposal facilities (Basnayake & Visvanathan, 2013). The collection, disposal, and recycling of solid waste pose

significant challenges in many nations, particularly in developing countries. In Sri Lanka, especially in Colombo, ineffective waste management has led to unclean and unhealthy urban environments. The Colombo Municipal Council bears the primary responsibility for maintaining the city's cleanliness (J, 1987).

However, logistical challenges, lack of infrastructure, and ineffective transportation routes contribute to inefficient waste disposal, leading to increased operational costs and environmental damage. In many developing countries, unplanned waste collection routes result in excess fuel consumption, traffic congestion, and higher CO_2 emissions. Thus, integrating route optimization into waste collection processes is crucial for achieving a more sustainable and environmentally friendly waste management system.

Route optimization involves applying mathematical modelling techniques such as Vehicle Routing Problem (VRP) algorithms, Dijkstra's Algorithm, and Mixed-Integer Linear Programming (MILP) to identify the most efficient waste collection routes. By utilizing graph theory and shortest path algorithms, these models minimize travel distances, reduce fuel consumption, and optimize fleet capacity. Additionally, metaheuristic approaches like Genetic Algorithms (GA) and Ant Colony Optimization (ACO) help solve large-scale, complex waste collection problems while balancing cost-effectiveness and sustainability.

Study focuses on optimizing waste collection and transportation routes in Kuppiyawatta East, through statistical analysis and mathematical modelling. Kuppiyawatta East, located in the Thimbirigasyaya Divisional Secretariat of Colombo District, is a densely populated urban area surrounded by Maligawatta East, Maligawatta West, Maligakanda, Borella North, Wanathamulla, and Kuppiyawatta West. It houses key institutions such as St. John's College, Nalanda College, and Ananda College (Kuppiyawatta East Grama Niladhari Division, n.d.) and faces significant waste management challenges due to its high population density and urban congestion (Wikipedia Contributors, 2024).

This study aligns with Sustainable Development Goal (SDG) 13: Climate Action, In the context of SDG 13 and combating climate change, the term defined in the United Nations Framework Convention on Climate Change (UNFCCC) (1992), as the 'change of climate which is attributed directly or indirectly to human activity that alters the composition of the global atmosphere and

which is in addition to natural climate Key Terms SDG 13: Take urgent action to combat climate change and its impact(*A LEGAL GUIDE*, n.d.).

Waste management and climate action are deeply interconnected, playing a crucial role in achieving SDG 13, Target 13.5, which emphasizes promoting mechanisms for raising capacity for effective climate change related planning and management. Optimizing waste collection routes through mathematical modelling enhances efficiency, reduces fuel consumption, and lowers CO_2 and methane emissions. Strengthening waste management systems with data-driven strategies supports climate resilience, minimizes environmental impact, and contributes to a sustainable future.

Problem Statement

Colombo, Sri Lanka's capital, is grappling with severe challenges in Municipal Solid Waste Management (MSWM) due to rapid urbanization and population growth. The city generates approximately 800 metric tons (MT) of solid waste daily, of which only about 450-500 MT of degradable waste and 150-200 MT of non-degradable waste are collected, resulting in a collection coverage of approximately 87.5%. The primary disposal methods include open dumping and composting, which are increasingly inadequate for the city's growing waste output. A significant portion of MSWM costs in Colombo is allocated to waste collection and transportation rather than disposal and treatment. The current practice involves transporting all waste from households to disposal stations along predetermined routes. This process is not only labour-intensive but also incurs high fuel and maintenance costs, contributing to inefficiencies and environmental degradation.

The core issue lies in the lack of an optimized vehicle routing system for waste collection and transportation. Inefficient routing results in increased operational costs, higher emissions, and extended collection times, further exacerbating the challenges faced by municipal authorities. Improving the routing system could minimize the total distance traveled, reduce costs, and enhance the overall efficiency of the waste collection process. Furthermore, the problem is compounded by the absence of adequate infrastructure for waste processing and disposal, alongside public resistance to proper waste management practices. Optimizing vehicle routing is crucial for

addressing these challenges and achieving a more sustainable and cost-effective waste management system in Colombo.

Research Objectives

1. To examine the waste collection routes currently used in Kuppiyawatha East and assess inefficiencies in terms of travel distance and operational constraints due to the absence of structured optimization.
2. To develop an optimized waste transportation model using Dijkstra's Algorithm**,** minimizing the total distance traveled while ensuring all waste collection points in Kuppiyawatha East are covered efficiently.
3. To evaluate the improvements in travel distance and route efficiency achieved through the optimized model compared to an unoptimized approach.
4. To fulfil the SDG 13 (climate action) target 13.5 - promote mechanisms to raise capacity for planning and management.

2. Literature Review

Efficient management of solid waste remains a big challenge in most urban areas, especially in the Colombo area. Different studies have been done by several researchers over the years to find ways of improving waste collection systems. Route optimization has become one of the key focus areas, which could help in improving efficiency, reducing operational costs, and decreasing environmental and climatic impacts. This review will help us to identify the local and global findings on route optimization techniques in solid waste management, focusing on different approaches used to handle the challenges faced.

A study conducted in the Ratmalana area aimed at optimizing the MSW collection and transportation process to address the high operational costs. In this regard, the application of linear programming techniques has helped minimize the total distance of transportation along with fuel consumption and emission. Based on the secondary data pertaining to September 2020, the study revealed that optimization would reduce the daily travel distance by 31.23 km and thus would be cost-effective (R & MDN, 2022).

A study focused on Gampaha, the second most densely populated district in Sri Lanka, modified the maximum flow amount technique to optimize municipal solid waste (MSW) collection via the shortest path model. With the use of GIS tools and Google Maps, the researchers determined the optimum route, which reduced the daily travel distance to 858 km, representing more than a 10% enhancement; furthermore, the vehicle requirement was reduced from 10 to 8, further reducing the distance by 14.2% and vehicle allocation by 20%. These optimizations have created room for considerable cost savings, especially in fuel consumption, establishing the foreplay of mathematical modelling in waste management (Hakmanage & D.D.M. Jayasundara, 2018).

Another case study on the optimization of MSW collection by routed tractors and assigning road segments in Kurunegala involved a developed methodology. By making use of the Capacitated Arc Routing Problem and Solver Studio through the use of Binary Integer Programming, the researchers successfully reduced trips made by 19% every week and reduced traveled distance by up to 36%. The studies have shown that data-driven optimizations significantly enhance urban waste management system efficiency and contribute toward a reduction in operation costs, consumption of fuel, and carbon footprint (R. D. S. S. Rambandara et al., 2022).

The study, in Kampala, Uganda, found that by applying GIS tools in route optimization during waste collection, the travel distances and the number of trips were minimized, increasing cost savings. It further optimized the vehicle fleet capacity for efficient operation with reduced emissions and fuel consumption. Using the GIS tools, a new landfill was derived as the existing site neared its capacity. Travel distances are reduced, hence reducing costs and leaving the environment cleaner. Such an approach can be a great insight into effective waste management in cities like Colombo in Sri Lanka (Kinobe et al., 2015). These studies highlight the importance of route optimization in improving waste management efficiency. By applying these insights, we can develop adaptable strategies to enhance Colombo's waste collection system.

3. Methodology

Data Collection

Secondary data was collected from the Colombo Municipal Council website (https://www.colombo.mc.gov.lk/garbage-collection.php) for *District 3 Kuppiyawatte - East*, providing detailed information on road paths. Distances between selected roads were measured using these road paths, which were considered as nodes for the analysis. For reference, Node 1 is designated as the starting point, and Node 22 represents the landfill location. Google Maps was also used to verify and supplement distance data.

Table1: Node description

No. Node	Information
1	Siridhamma Mw
2	Ananda Rajakaruna Mw.
3	Baseline rd
4	J.D. Fernando Mw.
5	Carshoe garden
6	Campbel Terrace
7	Kittyakara Rd
8	Nalanda Place
9	Aliyas place
10	Sangikarama Rd
11	Punchi Borella Junction
12	Temple Rd
13	Mahindarama Mw
14	Karunarathne Abeysekara mw
15	Ketawalamulla lane
16	Wesly College
17	Nalanda Hostal
18	Sangamiththa Vidyalaya
19	Oldcents church
20	Nalanda College
21	Gothami Vidyalaya
22	KerawalaPiitiya

Table 2: Distance data between nodes

From	To	Distance (m)
1	2	750
1	3	1600
1	4	280
1	6	600
1	8	350
1	9	280
1	10	240
1	17	190
2	4	800
2	5	500
2	7	400
2	11	450
2	16	500
2	18	180
2	19	8
2	22	2700
3	10	1100
3	15	1400
3	17	1500
4	6	600
4	8	350
4	9	300
4	10	500
4	17	96
5	16	500
5	22	2700
6	7	270
6	11	350
6	18	180
7	8	600
7	18	400
7	19	450
8	9	300
8	17	270
9	10	500
9	12	400
9	13	550
9	14	550
9	15	400
9	17	200
9	20	300
9	21	500
10	15	500
11	18	170

12	13	190
12	14	160
12	15	550
12	21	140
13	20	300
16	22	2600
17	20	270
18	19	210
19	22	2700

Dijkstra's Algorithm for Route Optimization

Dijkstra's Algorithm is a well-established computational technique used to determine the shortest path between nodes in a weighted graph. Originally developed by Edsger W. Dijkstra in 1956, this algorithm has found extensive applications in route optimization due to its efficiency and adaptability. In graph-based systems, nodes represent specific locations, and edges represent the connections between them, weighted by parameters such as distance, time, or cost (Dijkstra, 1959). In this study, the algorithm is applied to optimize waste collection routes by determining the most efficient paths from waste generation points to landfill sites.

The algorithm relies on the principle of minimizing weights along graph edges. The shortest path is calculated using the equation:

$$D(v_i) = \min\left(D(v_i), D\left(v_j\right) + w\left(v_i, v_j\right)\right)$$

Here, $D(v_i)$ represents the shortest known distance from the source node to node v_i, $D\left(v_j\right)$ is the distance to an intermediate node v_j, and $w\left(v_i, v_j\right)$ is the weight of the edge connecting v_i and v_j (Santoso et al., 2024). Initially, all nodes are assigned a tentative distance of infinity, except the source node, which is set to zero. The algorithm iteratively updates the distances for each neighbouring node until all nodes have been processed.

The steps of Dijkstra's Algorithm are as follows:

1. Initialization: Assign an initial distance of infinity to all nodes except the source node, which is set to zero.

2. Node Selection: Identify the unvisited node with the smallest tentative distance and mark it as the current node.
3. Distance Calculation: For each neighbour of the current node, calculate the tentative distance through the current node. Update this distance if it is smaller than the previously recorded value.
4. Mark as Visited: Mark the current node as visited, ensuring it will not be revisited. Repeat Until Completion: Repeat the process until the shortest path to the destination node is finalized or all nodes have been visited.
5. Path Reconstruction: Backtrack from the destination node to the source node to reconstruct the optimal route (Santoso et al., 2024).

Implementation of Dijkstra's Algorithm Using Python

Python was utilized to implement Dijkstra's Algorithm for optimizing waste transportation routes due to its efficient computational capabilities and extensive library support. The implementation involved constructing a graph representation of collection points and road distances using the NetworkX library. The algorithm was executed using a priority queue (heap), ensuring that the shortest path was computed efficiently while dynamically updating distances. Additionally, Matplotlib was used to visualize the graph at each iteration, allowing for a step-by-step analysis of the path optimization process. This approach enabled the identification of an optimal route that minimizes travel distance while covering all required collection points, demonstrating Python's effectiveness in solving complex route optimization problems in waste management logistics.

4. Results and Discussion

The implementation of Dijkstra's Algorithm for route optimization successfully determined the shortest path covering all nodes while minimizing the total travel distance. The algorithm processed the data iteratively, expanding the shortest path at each step and updating distances accordingly. The final optimized route obtained was [1, 17, 10, 4, 9, 8, 20, 6, 12, 15, 2, 19, 13, 18, 21, 14, 7, 11, 5, 16, 3, 22]**,** with a total travel distance of 3450 meters**.** The following section provides a detailed discussion of the algorithm's step-by-step execution, demonstrating how the shortest path evolved through multiple iterations.

The following table summarizes the sequence of iterations, highlighting the node selected at each step, the shortest known distance at that point, and the updated shortest distances to neighbouring nodes.

Table 3: Iteration Breakdown of Dijkstra's Algorithm

Iteration	Current Node	Distance from Start	Updated Shortest Distances
1	1	0	{2: 750, 3: 1600, 4: 280, 6: 600, 8: 350, 9: 280, 10: 240, 17: 190}
2	17	190	{20: 460} (new update from Node 17)
3	10	240	{15: 680, 12: 680, 20: 460}
4	4	280	{3: 1340, 15: 740}
5	9	280	{} (No new updates)
6	8	350	{7: 950, 13: 830, 14: 830}
7	20	460	{} (No new updates)
8	6	600	{13: 760}
9	12	680	{18: 780}
10	15	680	{} (No new updates)
11	2	750	{} (No new updates)
12	19	758	{5: 1250, 16: 1250, 22: 3450}
13	13	760	{} (No new updates)
14	18	780	{} (No new updates)
15	21	780	{} (No new updates)
16	14	830	{} (No new updates)
17	7	870	{} (No new updates)
18	11	950	{} (No new updates)
19	5	1250	{} (No new updates)
20	16	1250	{} (No new updates)
21	3	1340	{} (No new updates)
22	22	3450	{} (Final node reached)

Algorithm Execution Process

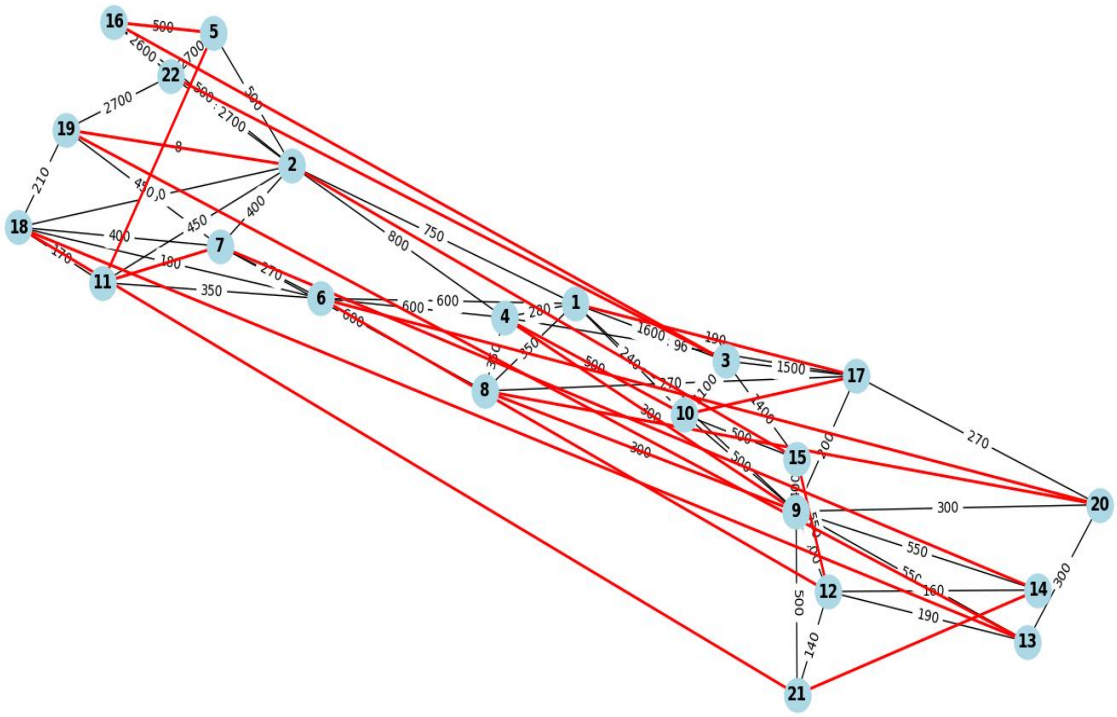
The algorithm began at Node 1**,** which was assigned a distance of 0, while all other nodes were initialized with a distance of infinity **(**`inf`**)**, as their shortest paths were unknown. A priority queue (min-heap) was used to select the next node to process, ensuring that the node with the smallest known distance was always selected first. In the first iteration, Node 1's directly connected neighbours were updated with their respective distances, marking the initial expansion of the shortest path.

As the algorithm progressed, it selected the nearest node at each step and updated the distances to its neighbouring nodes if a shorter path was found**.** Node 17 was the first to be processed after Node 1**,** as it had the smallest known distance of 190 meters. This led to an update in Node 20's shortest distance, which was recorded as 460 meters. Following this, Node 10 was selected in the next iteration with a distance of 240 meters, leading to further updates in the shortest known paths to Nodes 15 and 12, both set to 680 meters. The algorithm then proceeded to Node 4 and Node 9**,** each with a distance of 280 meters, refining paths to neighbouring nodes but not introducing significant changes beyond what was already established.

In the middle iterations, further refinements were made as Node 8 was selected with a distance of 350 meters, leading to updates in the shortest paths to Nodes 13 and 14, both set at 830 meters**.** Similarly, Node 6, processed at 600 meters, updated Node 13's shortest known path to 760 meters. At this stage, the algorithm had already determined paths for most of the nodes in the network. The later iterations primarily focused on confirming the shortest known paths rather than introducing major updates.

At Iteration 12, Node 19 was processed with a shortest distance of 758 meters, leading to updates in the shortest paths to Nodes 5, 16, and 22**.** This was a significant step because Node 22, the final destination, was now reachable with a total distance of 3450 meters**.** The algorithm then continued processing the remaining nodes, confirming distances but not making any significant refinements. The final iteration, Iteration 22**,** confirmed the total shortest distance as 3450 meters**,** marking the completion of the algorithm.

Figure 1: Graphical Representation of Iterations and Final Path



To better visualize the iterative process of Dijkstra's Algorithm, a graph is provided, illustrating how the shortest path evolved in each step. The graph depicts the network of nodes and connections, highlighting the nodes processed at each stage and the updated shortest paths.

The final optimized path obtained is clearly marked in the graph as: [1, 17, 10, 4, 9, 8, 20, 6, 12, 15, 2, 19, 13, 18, 21, 14, 7, 11, 5, 16, 3, 22]**.** This ensures that the shortest travel distance of 3450 meters is achieved while covering all required nodes.

Analysis of Algorithm Performance

The algorithm efficiently expanded the shortest known path in a systematic manner, always selecting the closest node first and refining paths accordingly. The priority queue (min-heap) played a crucial role in ensuring efficiency by always processing the node with the least distance,

reducing unnecessary computations. The early selection of Nodes 17, 10, and 4 significantly shaped the final route, as they provided crucial connections to multiple nodes in the network.

The total computed shortest distance of 3450 meters ensures that all nodes are covered with minimal travel distance. This confirms that the algorithm has successfully optimized the waste collection route while maintaining computational efficiency. The final sequence of node visits represents an optimal traversal path based on the given distance matrix.

5. Key Limitations and Future Research

Limitations

1. Lack of waste volume data: The research only had accumulated waste data for the entire district rather than individual collection points. This prevented the model from considering vehicle capacity constraints and load balancing.
2. Absence of operational cost data: The optimization was based purely on distance minimization, without factoring in fuel costs, labour expenses, or vehicle maintenance, limiting the model's real-world applicability.
3. No data on actual depot routes: The existing waste collection routes used by the depot in Kuppiyawatha East were unavailable, making it impossible to compare the optimized path with real-world practices.
4. Undirected graph assumption: The waste collection network was modeled as bidirectional, whereas real-world roads may have one-way restrictions, traffic conditions, and limited access zones that affect route optimization.

Future Research

1. Incorporating Waste Volume Data: Collecting real-time waste level data for each collection point would allow for dynamic vehicle load balancing, reducing unnecessary trips. This approach has been explored in studies utilizing ultrasonic sensors and IoT-based systems to monitor waste levels, enabling optimized collection routes (Long et al., 2019).

2. Cost Optimization: Future models should consider fuel consumption, labor costs, and maintenance expenses, using multi-objective optimization techniques to balance cost and distance efficiency. Research has demonstrated the effectiveness of integrating cost factors into route optimization for waste collection (Kumar et al., 2022).
3. Comparison with Real Depot Routes: By obtaining actual waste collection routes, future research can benchmark the optimized path against current operations to measure potential savings in time, distance, and cost. This comparison would provide practical insights into the benefits of the proposed optimization models.
4. Directed Graph Modeling: Implementing one-way road restrictions and road conditions in the model would improve its realism and accuracy. Considering road network constraints is crucial for developing feasible and efficient waste collection routes (Muthuvel et al., 2024).
5. Dynamic Routing Algorithms: Using real-time traffic and waste generation data, future models can implement adaptive route planning to respond to changing conditions. Studies have shown that dynamic route optimization can significantly enhance the efficiency of waste collection systems (Alharbi & Alshehri, 2020).
6. Exploring Alternative Optimization Techniques: Algorithms such as A*, Genetic Algorithms (GA), and Ant Colony Optimization (ACO) could be tested and compared against Dijkstra's Algorithm to find the best-performing approach for large-scale waste collection. Research indicates that these algorithms can offer advantages in solving complex routing problems (Karadimas et al., 2008).

6. Conclusion

This study successfully applied Dijkstra's Algorithm to optimize waste collection routes in Kuppiyawatha East, Colombo District, Sri Lanka, demonstrating its effectiveness in minimizing travel distances and improving operational efficiency. The algorithm computed an optimized path covering all collection points with a total travel distance of 3,450 meters, following the sequence of nodes representing waste collection points: [1, 17, 10, 4, 9, 8, 20, 6, 12, 15, 2, 19, 13, 18, 21,

14, 7, 11, 5, 16, 3, 22]. This optimized route significantly reduced unnecessary detours compared to unstructured routing methods. The results confirm that route optimization can enhance waste collection processes by lowering fuel consumption, reducing operational costs, and streamlining municipal waste transportation. Despite the success of the optimized model, certain limitations were identified. The study lacked real depot route data, preventing direct comparison with existing municipal collection practices. Additionally, the model was based solely on distance minimization, without incorporating factors such as vehicle capacity constraints, real-time traffic conditions, or operational costs. Addressing these limitations in future research by integrating cost-based route optimization, real-time waste tracking, and adaptive routing algorithms would further enhance the model's practical applicability. Nevertheless, this research provides a strong foundation for waste transportation optimization in Sri Lanka, demonstrating how graph-based algorithms can improve municipal waste collection systems. The findings emphasize the need for data-driven approaches in urban waste management, offering a roadmap for developing smarter, more efficient, and environmentally sustainable waste collection strategies in the country.

Reference

A LEGAL GUIDE. (n.d.). https://www.a4id.org/wp-content/uploads/2021/11/SDGLegalGuide_Chapter13_2022.pdf

Alharbi, W., & Alshehri, A. (2020). Dynamic route optimization for waste collection using genetic algorithm. In *Proceedings of the 2020 International Conference on Smart Internet of Things (ICOSICA)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICOSICA49951.2020.9243256

Basnayake, B. F. A., & Visvanathan, C. (2013). Solid Waste Management in Sri Lanka. *Municipal Solid Waste Management in Asia and the Pacific Islands*, 299–316. https://doi.org/10.1007/978-981-4451-73-4_15

Bayu Aji Santoso, Misbahu Surur, Syefudin Syefudin, & Gunawan, G. (2024). Application of dijkstra algorithm to optimize waste transportation distribution routes in Tegal Regency. *Jurnal Mandiri IT*, *13*(1), 81–90. https://doi.org/10.35335/mandiri.v13i1.290

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/bf01386390

EFL Admin. (2018). *Status of Waste Management in Sri Lanka*. Environment Foundation (Guarantee) Limited. https://efl.lk/status-waste-management-sri-lanka/

Hakmanage, N. M., & D.D.M. Jayasundara. (2018). Route optimization of solid waste collection in Gampaha. *Research Symposium on Pure and Applied Sciences, 2018 Faculty of Science*. https://www.researchgate.net/publication/344309429_Route_optimization_of_solid_waste_collection_in_Gampaha

J, D. S. (1987). *Major characteristics associated with the problem of solid waste management in the city of Colombo*. https://dl.nsf.gov.lk/handle/1/4955

Karadimas, N. V., Papatzelou, K., & Loumos, V. (2008). enetic Algorithms for Municipal Solid Waste Collection and Routing Optimization. In *Computational Intelligence in Complex Decision Systems* (pp. 223–246). Springer. https://doi.org/10.1007/978-0-387-74161-1_24

Kinobe, J. R., Bosona, T., Gebresenbet, G., Niwagaba, C. B., & Vinnerås, B. (2015). Optimization of waste collection and disposal in Kampala city. *Habitat International*, *49*, 126–137. https://doi.org/10.1016/j.habitatint.2015.05.025

Kumar, M., Singh, P., & Gupta, S. (2022). oute Optimization for Waste Collection. In *Proceedings of the International Conference on Recent Advances in Computational Techniques* (pp. 659–668). Springer. https://doi.org/10.1007/978-981-19-4193-1_59

Long, J., Gan, H., & Liu, X. (2019). imulation and Optimization of Dynamic Waste Collection Routes. *Waste Management & Research*, 37(8), 827–835. https://doi.org/10.1177/0734242X19833152

Muthuvel, P., Pandiyan, G., Manickam, S., & Rajesh, C. (2024). ptimizing Road Networks: A Graph-Based Analysis with Path-Finding and Learning Algorithms. *International Journal of Intelligent Transportation Systems Research*, 22(1), 123–135. https://doi.org/10.1007/s13177-024-00453-w

R, N. J., & MDN, N. G. (2022). Optimization of waste collection and transportation in Ratmalana area in Sri Lanka. *Proceedings of International Forestry and Environment Symposium*, *26*. https://doi.org/10.31357/fesympo.v26.5702

R. D. S. S. Rambandara, R. A. Ranga Prabodanie, E. A. C. P. Karunarathne, & R. D. D. Rajapaksha. (2022). Improving the Efficiency of Urban Waste Collection Using Optimization: a Case Study. *Process Integration and Optimization for Sustainability*, *6*(3). https://doi.org/10.1007/s41660-022-00232-8

Santoso, B. A., et al. (2024). Application of Dijkstra Algorithm to Optimize Waste Transportation Distribution Routes in Tegal Regency. Jurnal Mandiri IT, 13(1), 81–90. https://ejournal.isha.or.id/index.php/Mandiri/article/view/290

The importance of waste management | Prysmian Group. (n.d.). Www.prysmian.com. https://www.prysmian.com/en/insight/sustainability/the-importance-of-waste-management

United Nations Environment Programme. (2017, September 26). *Solid waste management*. UNEP - UN Environment Programme. https://www.unep.org/explore-topics/resource-efficiency/what-we-do/cities/solid-waste-management

Wikipedia Contributors. (2024, January 18). *Kuppiyawatta East Grama Niladhari Division*. Wikipedia; Wikimedia Foundation.

Appendix: Python Implementation of Dijkstra's Algorithm

A.1 Overview

This appendix contains the Python code used to implement Dijkstra's Algorithm for optimizing waste collection routes. The algorithm finds the shortest path covering all nodes while minimizing the total travel distance. The script includes:

- Graph Construction: Representing nodes and distances.
- Dijkstra's Algorithm: Finding the shortest path using a priority queue.
- Path Extraction: Retrieving the optimal route.
- Graph Visualization: Displaying the network and the computed path.

A.2 Python Code Implementation

```
import heapq
import networkx as nx
import matplotlib.pyplot as plt

# Step 1: Build the Graph Representation from Data
def build_graph(data):
    graph = {}
    for frm, to, dist in data:
        if frm not in graph:
            graph[frm] = {}
        if to not in graph:
            graph[to] = {}
        graph[frm][to] = dist
        graph[to][frm] = dist  # Since the graph is undirected
```

```
    return graph

# Step 2: Dijkstra's Algorithm with path tracking
def dijkstra(graph, start, end, all_nodes):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]
    previous_nodes = {node: None for node in graph}
    visited = set()
    path = []
    iterations = []

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_node in visited:
            continue

        visited.add(current_node)
        path.append(current_node)
        iterations.append((current_node, current_distance, dict(distances)))

        if len(visited) == len(all_nodes):
            break

        for neighbor, weight in graph[current_node].items():
            if neighbor not in visited:
                distance = current_distance + weight
                if distance < distances[neighbor]:
                    distances[neighbor] = distance
                    previous_nodes[neighbor] = current_node
                    heapq.heappush(priority_queue, (distance, neighbor))

    return distances, previous_nodes, path, iterations

# Step 3: Extract Shortest Path
def extract_path(previous_nodes, start, end):
    path = []
    current = end
    while current is not None:
        path.append(current)
        current = previous_nodes[current]
    path.reverse()
    return path

# Step 4: Visualization
def visualize_graph(graph, path=None):
    G = nx.Graph()
    for frm, neighbors in graph.items():
```

```
        for to, weight in neighbors.items():
            G.add_edge(frm, to, weight=weight)

    pos = nx.spring_layout(G)
    edge_labels = nx.get_edge_attributes(G, 'weight')

    nx.draw(G, pos, with_labels=True, node_size=500, node_color='lightblue',
font_weight='bold')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

    if path:
        path_edges = [(path[i], path[i + 1]) for i in range(len(path) - 1)]
        nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='red',
width=2)

    plt.show()

# Example Usage
data = [
    (1, 2, 750), (1, 3, 1600), (1, 4, 280), (1, 6, 600), (1, 8, 350), (1, 9,
280), (1, 10, 240), (1, 17, 190),
    (2, 4, 800), (2, 5, 500), (2, 7, 400), (2, 11, 450), (2, 16, 500), (2,
18, 180), (2, 19, 8), (2, 22, 2700),
    (3, 10, 1100), (3, 15, 1400), (3, 17, 1500), (4, 6, 600), (4, 8, 350),
(4, 9, 300), (4, 10, 500), (4, 17, 96),
    (5, 16, 500), (5, 22, 2700), (6, 7, 270), (6, 11, 350), (6, 18, 180), (7,
8, 600), (7, 18, 400), (7, 19, 450),
    (8, 9, 300), (8, 17, 270), (9, 10, 500), (9, 12, 400), (9, 13, 550), (9,
14, 550), (9, 15, 400), (9, 17, 200),
    (9, 20, 300), (9, 21, 500), (10, 15, 500), (11, 18, 170), (12, 13, 190),
(12, 14, 160), (12, 15, 550), (12, 21, 140),
    (13, 20, 300), (16, 22, 2600), (17, 20, 270), (18, 19, 210), (19, 22,
2700)
]

# Build the graph
graph = build_graph(data)

# Run Dijkstra's algorithm with all nodes to find the path passing through
all nodes
start_node = 1
end_node = 22
all_nodes = set(graph.keys())

distances, previous_nodes, path, iterations = dijkstra(graph, start_node,
end_node, all_nodes)

# Print results
if len(path) == len(all_nodes):
```

```
    print("Path covering all nodes:", path)
    print("Total Distance:", distances[end_node])
else:
    print("No valid path covering all nodes found.")

# Visualize the graph and the path
visualize_graph(graph, path=path if len(path) == len(all_nodes) else None)

# Print iteration states
for i, (node, dist, state) in enumerate(iterations):
    print(f"Iteration {i + 1}: Node {node}, Distance {dist}")
    print("State:", state)
```