OPTIMIZING THE ORDER DISTRIBUTION IN THE CITY OF TETOVO USING DIJKSTRA'S ALGORITHM

Miranda Xhaferi
University of Tetova, North Macedonia, miranda.xhaferi@unite.edu.mk
Elvir Iljazi
HotelKey Inc, United States of America, elvir.iljazi@hotelkeyapp.com

Abstract: The major concerns of e-commerce are the optimization of order dispatch operations and delivery time prediction. Finding the optimal path is considered an important problem in logistics systems. Efficient order distribution is crucial for enhancing logistic operations, especially in urban environments like the city of Tetovo. Therefore, proper prediction and optimization for delivery operations are required for optimal logistics management. This study proposes a novel approach to optimize order distribution utilizing Dijkstra's Algorithm, a well-established method in graph theory and network analysis. To ensure better visibility of the logistics activities and avoid possible obstacles it is suitable to adopt a graph-based method. By modeling Tetovo city's road network as a graph, with intersections as nodes and roads as edges, the algorithm calculates the shortest paths between various distribution points, such as warehouses and delivery destinations. From the drop points, we will get the information and data about the delivery address, weight of goods packages, and number of customer packages carried by a courier in one delivery transfer by truck delivery logistics. In the meantime, the data about the distance from the distribution points to the delivery destinations will be obtained from the Open Street Maps applications. All the data above will be used to construct a connected weight graph as an initial model graph. Using a methodology based on Dijkstra's Algorithm, we will determine the shortest or fastest route for delivery service in logistics distribution. By denoting the location of a drop point as a starting point of the route and the road connecting two locations as an edge where the distance traveled from the drop point to the customers or from one customer to another will represent the weight in the graph, we can apply the Dijkstra's algorithm to optimize the mileage of the delivery packages. The optimization process aims to reduce delivery time and minimize costs while maximizing resource utilization and customer satisfaction. The implementation of Dijkstra's Algorithm in this context involves factors like traffic congestion, road conditions, and delivery priorities. This algorithm works by finding the lowest cost path with various costs associated with the edges enabling the most efficient way of completing an order within a factory or warehouse. We represent a JavaScript implementation of this algorithm. Through simulation and empirical validation, the proposed approach demonstrates significant improvements in order distribution efficiency compared to traditional methods. This research contributes to the advancement of logistic optimization techniques and provides practical insights for enhancing urban delivery systems in the city of Tetovo and similar metropolitan areas.
Keywords: Dijkstras's Algorithm, order distribution, optimization, graph, road network.

1. INTRODUCTION

The growing technological development has offered advanced solutions to ease people's life such as online shopping (Wu, 2013; Zhang et. al. 2019). The e-commerce managers, nevertheless, depend on logistics firms to guarantee the delivery of orders, giving customers the flexibility to select delivery options based on factors such as distance and waiting time. The major concerns of e-commerce are the optimization of order dispatch operations and delivery time prediction. Finding the optimal path is considered an important problem in logistics systems. Efficient order distribution is crucial for enhancing logistic operations, especially in urban environments like the city of Tetovo. Therefore, proper prediction and optimization for delivery operations are required for optimal logistics management.

It has been established that any system can be represented as a collection of nodes, where certain pairs of nodes are linked by particular relationships represented by lines. This concept gave rise to the development of graph theory. The motivation behind the creation of this theory and the curiosity surrounding it arises from its broad applicability across various fields.

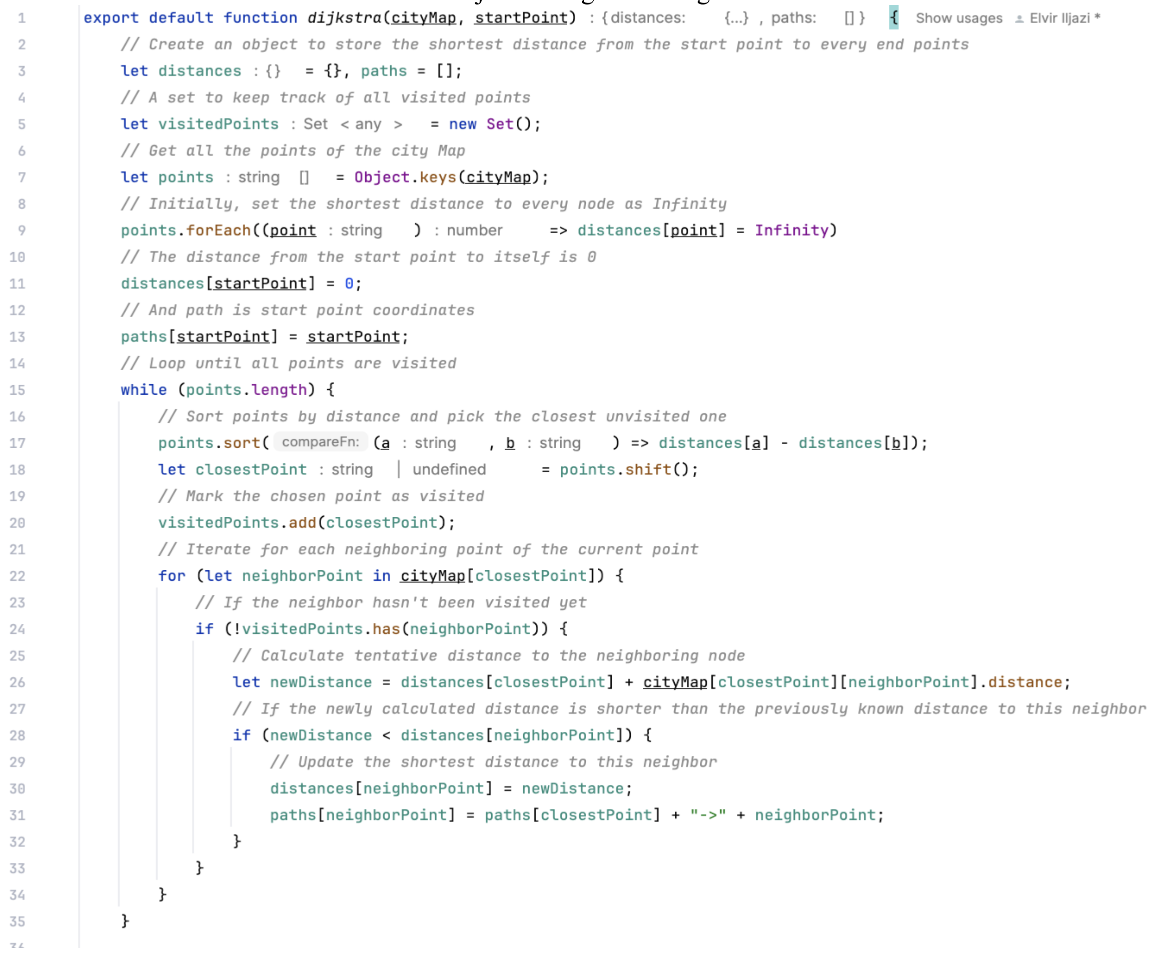
Due to the extensive range of problems addressed in graph theory, it has played and continues to play a significant role in addressing numerous issues. The concept of finding the shortest path is recognized as a crucial aspect of graph theory. Graph theory and network analysis are considered as one of the most important topics regarding operations research. Research has been continuously conducted to identify optimal algorithms for addressing the challenge of determining the shortest path. The application of graph theory to various real-world scenarios is straightforward. When dealing with the shortest path algorithm, the analysis centers on two nodes or vertices along the path to identify the most optimal solution for the shortest route.

Through our research, we have utilized OpenStreetMap and the Leaflet library to implement Dijkstra’s Algorithm, a well-established method in graph theory and network analysis and a powerful tool for finding the shortest paths in a network. To ensure better visibility of the logistics activities and avoid possible obstacles it is suitable to adopt a graph-based method. By modeling Tetovo city’s road network as a graph, with intersections as nodes and roads as edges, the algorithm calculates the shortest paths between various distribution points, such as warehouses and delivery destinations.

2. DIJKSTRA’S ALGORITHM

Dijkstra's algorithm, developed by E.W. Dijkstra, is a method for finding the shortest path in a given graph (Morris, 2016) (Zhang et. al., 2005), used to solve the single-source shortest-path problem when all edges have non-negative weights.
In a graph, the algorithm starts at the initial node and grows a tree that ultimately spans all nodes that can be accessed from the initial node.
The algorithm operates in an iterative manner, in which at each iteration, it selects the node with the shortest distance path from the initial node and recalculates the path distance for the remaining unvisited nodes.
This process will be repeated until the destination node is visited (Zhang et. al., 2005).
One of the most important advantages of Dijkstra’s algorithm is that it does not visit the remaining unwanted nodes when the intended destination node is reached (Chan et. al., 2016).

Code of Dijkstra’s algorithm is given below:



```
export default function dijkstra(cityMap, startPoint) : {distances: {...} , paths: [] } {  Show usages  Elvir Iljazi *
    // Create an object to store the shortest distance from the start point to every end points
    let distances : {}  = {}, paths = [];
    // A set to keep track of all visited points
    let visitedPoints : Set < any >  = new Set();
    // Get all the points of the city Map
    let points : string  []  = Object.keys(cityMap);
    // Initially, set the shortest distance to every node as Infinity
    points.forEach((point : string  ) : number    => distances[point] = Infinity)
    // The distance from the start point to itself is 0
    distances[startPoint] = 0;
    // And path is start point coordinates
    paths[startPoint] = startPoint;
    // Loop until all points are visited
    while (points.length) {
        // Sort points by distance and pick the closest unvisited one
        points.sort( compareFn: (a : string  , b : string  ) => distances[a] - distances[b]);
        let closestPoint : string | undefined    = points.shift();
        // Mark the chosen point as visited
        visitedPoints.add(closestPoint);
        // Iterate for each neighboring point of the current point
        for (let neighborPoint in cityMap[closestPoint]) {
            // If the neighbor hasn't been visited yet
            if (!visitedPoints.has(neighborPoint)) {
                // Calculate tentative distance to the neighboring node
                let newDistance = distances[closestPoint] + cityMap[closestPoint][neighborPoint].distance;
                // If the newly calculated distance is shorter than the previously known distance to this neighbor
                if (newDistance < distances[neighborPoint]) {
                    // Update the shortest distance to this neighbor
                    distances[neighborPoint] = newDistance;
                    paths[neighborPoint] = paths[closestPoint] + "->" + neighborPoint;
                }
            }
        }
    }
```

Source: From https://github.com/elviriljazi/dijkstra-in-tetovo/blob/main/utilities/dijkstra.js

On the other hand, the disadvantage of Dijkstra’s algorithm lies in its challenging implementation in computer programs when dealing with a large number of nodes, as this can lead to significant consumption of CPU memory during computation. (Aghaei et. al., 2009).

The primary objective of the shortest path algorithm is to provide quick responses even with extensive input graphs. Dijkstra's algorithm is utilized to efficiently determine the shortest path between two vertices. The implementation of JavaScript programming language is essential for identifying the shortest path and meeting related criteria. This program, known as the shortest path optimization system, is further explained through a Unified Modeling Language (UML) class diagram using JavaScript. It is important to note that the Dijkstra algorithm is limited to positive weight graphs and cannot accommodate negative edges, with the exact path remaining unknown.

3. PROBLEM STATEMENT

During the dispatching process, the identification of the most optimal route remains a notable issue, potentially originating from different real-world situations. Logistics companies must optimize their operations, particularly in the delivery of goods, to reduce waiting time and eliminate unnecessary tasks, consequently improving customer satisfaction. In a dispatching problem within an urban area, nodes denote locations where an order can be transferred to other points along a specific route. In this specific context, the edges within the graph represent the roads that link two nodes. The weights assigned to these edges typically denote factors such as distance, the cost of transportation, the time, etc.

In this research, the experiment has been conducted utilizing a specialized JavaScript application.

The study was designed to evaluate the proposed algorithm under a variety of conditions, including the use of large versus small sample data, long versus short travel distances, and different numbers of generations in algorithm implementation.

To ensure precision in the results, the algorithm was executed 20 times for each experiment case, followed by the computation of the average value as the conclusive outcome. The comparison between large and small data sets involved testing the algorithm with the same start and end points but utilizing distinct data sets.

In our study, we drew inspiration from the city of Tetovo as an exemplar for modeling our logistics network. Tetovo, located in the western part of North Macedonia, offers a diverse urban landscape characterized by various transportation challenges and logistical intricacies. By leveraging Tetovo as a reference point, we aimed to capture the complexities of real-world logistics scenarios within our simulation.

By representing the road network of Tetovo city as a graph where intersections serve as nodes and roads as edges, the algorithm computes the most efficient routes connecting different points of distribution, such as warehouses and delivery locations.

The logistics network we developed closely mirrors the road infrastructure and geographical layout of Tetovo, incorporating 8,666 roads interconnecting 9,681 points. This emulation of Tetovo's urban environment enables us to simulate and analyze logistics operations within a context that closely resembles real-world conditions.

Drawing from Tetovo as a case study provides valuable insights into the challenges and opportunities inherent in urban logistics management. By aligning our simulation with the characteristics of Tetovo, we can derive practical implications and recommendations for improving logistics efficiency in similar urban settings.

The utilization of Tetovo as an example underscores the applicability and relevance of our research findings to urban logistics management practices. This approach enriches the authenticity and validity of our study, enhancing its potential impact on the field of logistics optimization.

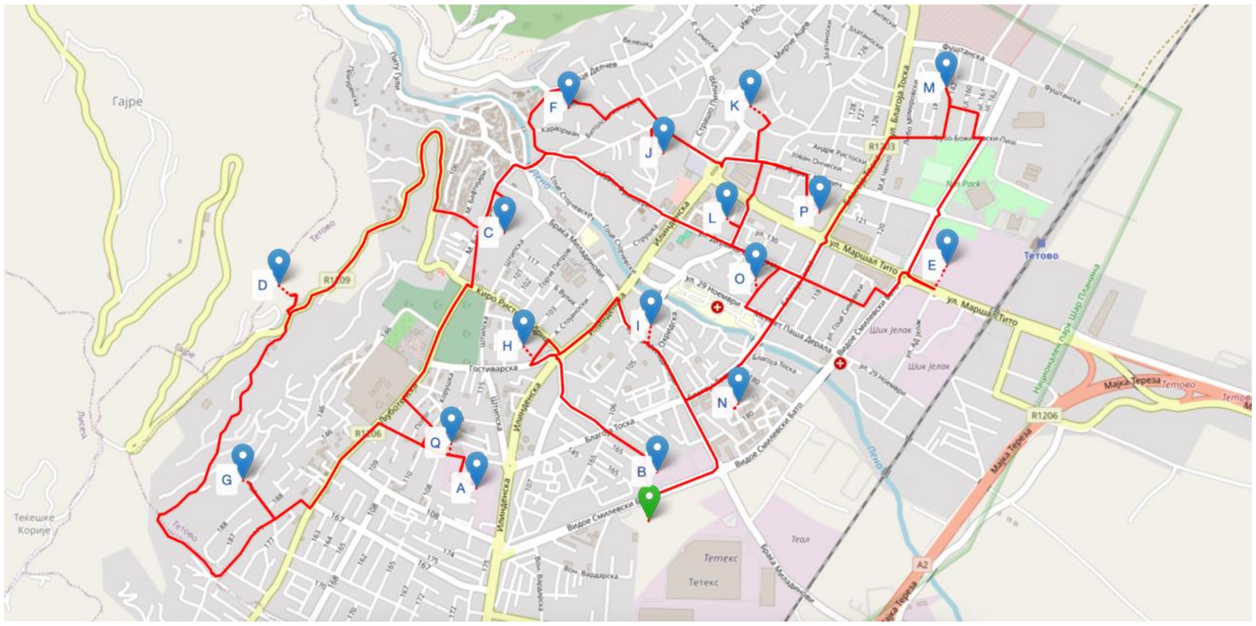
Incorporating insights gleaned from Tetovo into our research paper elucidates the contextual basis of our simulation and underscores the practical significance of our findings within the realm of urban logistics management.

The model being studied is associated with a package delivery service conducted by a courier
starting from SabahCargo drop-off point located at Vidoe Smilevski Bato. From the drop points, we will get the information and data about the delivery address, weight of goods packages, and number of customer packages carried by a courier in one delivery transfer by truck delivery logistics.

The courier serves more than 15 clients on a single delivery trip, depending on demand density. Specifically, the analysis focuses on serving 17 customers across 17 locations in Tetovo City regions. In the meantime, the data about the distance from the distribution points to the delivery destinations will be obtained from the Open Street Maps applications. Utilizing this data, an initial model graph is developed, representing a connected weighted graph. Here, each drop-off point or customer location serves as a vertex, while the roads between locations act as edges in the graph. The weight assigned in this graph signifies the distance covered from the drop-off point to customers or between different customers. Then the Dijkstra algorithm is applied to the graph with the drop point as the starting point, ensuring that the courier visits all customers and returns to the drop point. The research methodology flow begins with the collection of customer location data, followed by the conversion of location data into distance data between locations. Afterward, this data is employed for the creation of an initial model which is shown as a connected weighted graph. The Dijkstra algorithm is then applied to this graph to determine the shortest path that covers all clients and returns to the initial drop-off point.

This route can be a closed path (cycle) or a closed trail (circuit) within the initial model graph.
The construction of the initial model graph involves data on customer locations, road connectivity, road conditions, and distances between locations, with roads suitable for passage. In cases where two locations are not directly connected by a road, no edge is present between them in the graph. The model graph, denoted as G, is a connected weighted graph.
Figure 1. presents a visual depiction of the logistics network inspired by infrastructure and geographical layout of Tetovo. The green point on the map represents the dispatching center, designated as the focal point for logistics operations. Surrounding the dispatching center are 10 customer nodes, symbolizing various delivery destinations within the urban area. Each node is strategically positioned to reflect the distribution of orders across Tetovo. The visualization provides a comprehensive overview of the logistics network, showcasing the connectivity between the dispatching center and customer nodes via the road network.

Figure 1. The subgraph G' with an example route.



Source: From https://dijkstra-in-tetovo.vercel.app/

4. SIMULATION RESULTS AND ANALYSIS

This section discusses the findings derived from implementing the algorithm simulations. Dijkstra's algorithm is used to determine the solution for single-source, single-destination, and single-pair shortest-path problems. Notably, the Dijkstra algorithm is particularly effective in scenarios with positive weights within directed or undirected graphs, especially when the graph contains a substantial number of nodes. It is worth mentioning that the performance of the Dijkstra algorithm improves when dealing with a large number of nodes.
Our findings align with those of previous studies. For instance, (Singh & Tripathi, 2018), demonstrated that Dijkstra is highly effective across a wide range of nodes. Similarly, (Sapundzhi & Popstoilov, 2018) found that for a larger number of nodes, Dijkstra's algorithm exhibits superior efficiency. Furthermore, (Lacorte & Chavez, 2018), observed that Dijkstra's algorithm consistently yields shorter processing times for graphs of various sizes.
We conducted computational experiments to evaluate the performance of the Dijkstra algorithm in optimizing logistics operations within this simulated network. The results of these experiments, including the calculation time for each route, are summarized in the table below:

Table 1. Average running time of the algorithm

Nodes	Execution Time(ms)
5	5327
10	9801
15	14102
20	18293

Source: Adapted from Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization by Samah et.al., 2020, https://www.researchgate.net/publication/345332893_Comparative_Analysis_between_Dijkstra_and_Bellman-Ford_Algorithms_in_Shortest_Path_Optimization

5. CONCLUSIONS

By utilizing simulation and empirical validation, the proposed method showcases notable enhancements in the effectiveness of order distribution processes when compared to traditional methods. For the simulations that were utilized, we developed a web application https://dijkstra-in-tetovo.vercel.app/ .

This research contributes to the advancement of logistic optimization techniques and provides practical insights for enhancing urban delivery systems in the city of Tetovo and similar metropolitan areas and will be an open source for anyone interested in this algorithm in future development and research. Our project aims to contribute valuable insights to the field of logistics and transportation planning. By applying Dijkstra's Algorithm to the unique urban landscape of Tetovo, we have identified strategies to elevate order distribution procedures, reduce delivery times, and minimize resource wastage. It is worth mentioning that the performance of the Dijkstra algorithm improves when dealing with a large number of nodes.

REFERENCES

Aghaei, M.R.S., Zukarnain, Z.A., Mamat, A., & Zainuddin, H. (2009). "A Hybrid Algorithm for Finding Shortest Path in Network Routing", Journal of Theoretical and Applied Information Technology.

Beker, I., Jevtic, V., & Dobrilovic, D. (2012). "Shortest-path algorithm as a tools for inner transportation optimization", International Journal of Industrial Engineering and Management (IJIEM), Vol.3 No 1, pp. 39-45.

Chan, S., Adnan, N., Sukri, S.S., & Zainon, W.M. (2016). An experiment on the performance of shortest path algorithm.Knowledge Management International Conference (KMICe) 2016, 29 – 30 August 2016, Chiang Mai, Thailand.

Lacorte, A. M., & Chavez, E. P. (2018). Analysis on the Use of A* and Dijkstra's Algorithms for Intelligent School Transport Route Optimization System. Proceedings of the 4th International Conference on Human-Computer Interaction and User Experience in Indonesia, CHIuXiD '18 - CHIuXiD '18. doi:10.1145/3205946.3205948.

Lusiani, A., Samsiyah, S., & Sartika, P.E. (2023). DIJKSTRA ALGORITHM IN DETERMINING THE SHORTEST ROUTE FOR DELIVERY SERVICE BY J&T EXPRESS IN BANDUNG. Lebesgue: Jurnal Ilmiah Pendidikan Matematika, Matematika dan Statistika. Vol. 4, No. 2.

Morris, J., (2016, February 15) 10.2 Dijkstra's Algorithm. Available from: https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html.

Samah, W.G., Salim, A. , Ibrahim, R., Saringat, M. Z., Jamel, S., & Wahab, J.A. (2020). Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization. IOP Conference Series: Materials Science and Engineering, Volume 917, International Conference on Technology, Engineering and Sciences (ICTES). Penang, Malaysia.

Sapundzhi, F. I., & Popstoilov, M. S. (2018). Optimization algorithms for finding the shortest paths. Bulgarian Chemical Communications, Volume 50, Special Issue B, (pp. 115 – 120).

Singh, J.B., & Tripathi, R.C. (2018). Investigation of Bellman–Ford Algorithm, Dijkstra's Algorithm for suitability of SPP. IJEDR , Volume 6, Issue 1, ISSN: 2321-9939 .

Wu, I. L. (2013). The antecedents of customer satisfaction and its link to complaint intentions in online shopping: An integration of justice, technology, and trust. *Int. J. Inf. Manage.*, vol. 33, no. 1, pp. 166–176.

Zhang, F., Qiu, A., & Li, Q. (2005). Improve on Dijkstra Shortest Path Algorithm for Huge Data. Chinese academy of surveying and mapping.

Zhang, N., Yang, Y., Zheng, Y., & Su, J. (2019) . Module partition of complex mechanical products based on weighted complex networks. *J. Intell. Manuf.*, vol. 30, no. 4, pp. 1973–1998.