Journal of Computer Science

Original Research Paper

Cargo Route Optimization Using Shortest Path Algorithms: Runtime and Validity Comparison

Dedy Rahman Wijaya, Aqil Athallah, Tiara Nuha Noor'afina, Patrick Adolf Telnoni and Sari Dewi Budiwati

School of Applied Science, Telkom University, Bandung, Indonesia

Article history
Received: 15-05-2023
Revised: 27-07-2023
Accepted: 04-09-2023

Corresponding Author:
Dedy Rahman Wijaya
School of Applied Science,
Telkom University Bandung,
Indonesia
Email: dedyrw@telkomuniversity.ac.id

Abstract: The Indonesian logistics industry is facing significant challenges related to inefficiency and irregularity, particularly in the commodity cargo route system. This issue is further exacerbated by the high logistics costs in the country, which currently stands at 23%, higher than that of other countries. To address this issue, this study proposes an implementation and examination of several algorithms Greedy, Best First Search (BFS), Dijkstra's, A*, and Floyd-Warshall to optimize the cargo route system and reduce logistics costs. The algorithms were compared using various parameters, including price, distance, rating, and time. The results revealed that the Greedy algorithm is not a reliable option for cargo route optimization. In contrast, the A* algorithm offered the best solution compared to other algorithms, although it was not the fastest in terms of time. This study emphasizes the importance of considering various factors to optimize the cargo route system effectively. The experiments conducted in this study offer promising insights and pave the way for further research to improve the efficiency and reliability of the logistics industry in Indonesia.

Keywords: Cargo Route, Shortest Path, Greedy, A*, Dijkstra's, Floyd-Warshall, Best First Search

Introduction

The presence of transportation and logistics infrastructure has been crucial in enhancing the economy of Indonesia, particularly during the new normal era when mobility was restricted to adhere to health protocols and strict supervision (Widiyanto, 2020). Logistics efficiency is a crucial factor in developing inter-regional trade cooperation. In many areas in eastern Indonesia, the inadequate logistics system results in a low level of competitiveness. The limited cargo volume indicates the small economies of scale for these islands, consequently leading to high transportation costs (Amin *et al*., 2021). Logistics activities in Indonesia still have problems with inefficiency and irregularity in the logistics system, especially the commodity cargo route system. Logistics costs in Indonesia are generally high compared to other countries, the ratio is quite large at 23% (Kargo, 2021). Many countries have implemented digitalization during the 4.0 industrial revolution in their logistics activities. Several companies have started implementing digitalization. Artificial Intelligence (AI) is one of the crucial technologies that cannot be overlooked in the context of the 4.0 industrial revolution. Examples of technology adoption and digital automation in the use of AI technology for logistics service work, which is optimizing delivery routes (Raza *et al*., 2020). Therefore, the application of AI technology in Indonesia's logistics system is necessary. One application of AI technology is to optimize delivery routes by finding the shortest path.

The shortest path algorithm is an essential tool for finding the most efficient route between two or more nodes in a graph. The algorithms are used in several applications, namely, traffic routing, parcel delivery, public transportation, and disaster response (Liu *et al*., 2018). In traffic routing, the algorithm was used to identify the fastest route between multiple points. The road capacity, traffic volume, and speed limits can be considered to identify the shortest path (Shan *et al*., 2018). Moreover, traffic routing could help to find the fastest route in the event of a natural disaster. Thus, the

application could minimize the disaster impact (Liu *et al*., 2018). In parcel delivery, the algorithm was used to minimize fuel consumption and delivery time. As a result, the carbon footprint was reduced (Gbadamosi and Aremu, 2020). Whereas in public transportation, the algorithm was planned to find the most efficient routes. Therefore, travel time efficiency and reliability were improved (Cheng *et al*., 2019). The cargo route application generates the shortest path based on several parameters such as price, distance, rating, and time. In distribution management, determining schedules and shipping routes from one point to multiple destinations is crucial for companies to minimize shipping costs. However, transportation capacity and delivery deadlines are other important factors that need to be considered in the distribution process.

The current study is not a new problem. This research does not introduce a new method but rather conducts a unique comparative study. The authors investigate and compare existing methods to evaluate their effectiveness and relative advantages in a specific context. Thus, our research provides new insights into the existing approaches and identifies strengths and weaknesses that can be derived from these approaches.

Many examples of shortest-path algorithms found online are often presented in the form of pseudo-code or program fragments that require further work to be implemented in a specific programming language or data structure. However, many sources provide implementations of these algorithms in various programming languages and data structures. One example that was found is Dijkstra's algorithm, which is often presented in pseudo-code that can be seen in the sub-section "Proposed Method: Dijkstra Algorithm." There is also a website page that provides implementations of shortest-path algorithms in various programming languages and data structures (https://www.geeksforgeeks.org/).

In this study, a fresh method for route optimization is presented, employing a combination of five distinct algorithms to search for the shortest path. These shortest path search algorithms consider cargo route data, distance, price, time, and service ratings to dynamically adapt and find the most efficient routes. Different from previous studies that implement one or two algorithms for comparison, our research implements five traditional algorithms to find the algorithm that provides the best cargo route, especially in Indonesia. Therefore, based on the previous explanation, the contribution of this study can be explained as follows:

1. Standard implementation of shortest path algorithms has been developed based on Python programming language and Representational State Transfer Application Programming Interface (REST API) such as Greedy, Best First Search (BFS), Dijkstra's, A*, and Floyd-Warshall (Wijaya, 2023)
2. The performance investigation of several shortest path algorithms for cargo route optimization uses several parameters: Price, distance, rating, and time. Their performance is investigated based on runtime and validity

Related Works

Currently, researchers or practitioners often require examples of algorithms, especially for finding the best route. Most examples of shortest-path algorithms are only in the form of pseudo-code or code snippets that may not always be executable. Many sources on the internet only implement one algorithm in various programming languages.

To address this issue, our research focuses on comparing the results of five shortest path algorithms based on runtime and validity to determine which search algorithm is the fastest and most accurate in determining the shortest route.

The shortest path is defined as the minimum number of weight or distance values that must be traversed to reach the destination point from the starting point in a network or graph. There are several studies related to path-finding algorithms and including them can explain the position and importance of our research.

A case study utilizing the Genetic algorithm involved optimizing the sequence of routes for a cargo truck, starting from the first node and continuing to the subsequent nodes until reaching the destination. It resulted in 480 best generations and identified the shortest total distance. However, the Genetic algorithm has a significant drawback, namely the long computer time and burden required to reproduce and generate new candidate solutions until it achieves the goal of the problem (Evangelista *et al*., 2020).

Greedy and Floyd-Warshall algorithms are two other algorithms used to determine the shortest path. The Greedy algorithm operates by making locally optimal choices at each step in the hope of finding a globally optimum solution. Meanwhile, the Floyd-Warshall algorithm is simpler and easier to implement. The Floyd-Warshall algorithm considers all nodes and routes to find the best result (Azis *et al*., 2018a). This literature indicates that while Dijkstra's algorithm is generally faster for most graph classes, the Floyd-Warshall algorithm can be competitive in some cases due to its ability to solve dense linear systems and perform matrix multiplication (Sao *et al*., 2020).

Dijkstra's is one of the algorithms that can be modified to produce the shortest path (Gbadamosi and Aremu, 2020). Despite the fact that Dijkstra's algorithm does not always guarantee to find the globally optimal solution, it can still provide a reasonably optimal solution, and in

certain cases, the globally optimal solution can be obtained with some probability (Qian and Yinfa, 2018). In addition to Dijkstra's, the A* algorithm can find the shortest path more efficiently than other algorithms (Ju *et al*., 2020). A* algorithm's ability to find the shortest path with a heuristic function makes it a suitable choice for generating personalized route recommendations in PRR tasks through machine learning and neural networks (Wang *et al*., 2021). Additionally, there is also the Best First Search (BFS) algorithm, which is a heuristic search algorithm that combines the advantages of both breadth-first search and depth-first search by taking the strengths of each algorithm (Liana and Nudin, 2020).

A combination modification of Chemical Reaction Optimization (CRO) and the BFS algorithm can also produce high-quality shortest paths compared to the CRO metaheuristic algorithm (Khattab *et al*., 2022). Path-finding algorithms can also be used to efficiently calculate the number of shortest paths in large highway networks (Qiu *et al*., 2022). Developing the shortest path-finding algorithms can help optimize transportation infrastructure systems to find the best routes between origin and destination (Shan *et al*., 2018). The GraphX shortest path algorithm's efficiency in finding the shortest path is limited as it only considers the number of edges without using their weights, resulting in a longer computation time compared to other algorithms (Phan and Do, 2018).

There is a study on traditional navigation systems that compute the quantitatively shortest or fastest route between two locations in a spatial network. This study used the k-shortest path algorithm to find vehicle routes in Washington DC using a single parameter and encountered some failures when attempting to find routes, but its runtime was still reasonable (Cheng *et al*., 2019). The traditional method for calculating the shortest path distance is no longer suitable for large graphs with millions of nodes and billions of edges (Rizi *et al*., 2018). Nevertheless, the traditional method can still be used and performs well in determining the shortest routes. The authors also utilized several sources from the journal of computer science to strengthen their research. First, a study finding the shortest path for public transportation to solve single-source using Dijkstra's algorithm (Wongso *et al*., 2018) and calculate the most efficient route for COVID-19 transmission within extensive community networks, aiming to analyze and forecast the progression of the transmission chain (Mavakala *et al*., 2023).

Materials and Methods

Materials

To build the graph data structure in Python, a dictionary can be used (Ireland and Martin, 2020). The dictionary data structure is commonly used to represent graphs, where each node is represented as a key in the dictionary, and its corresponding value is a list or set containing the neighboring nodes or edges connected to that node. The keys in the dictionary represent the nodes, while the values represent the connections or edges between nodes. Cost is calculated based on several parameters, such as price, distance, rating, and time. Several shortest path algorithms are implemented to find the best route. Finally, performance of algorithms is evaluated based on runtime and validity. The authors evaluated the algorithm with several jump (hop) values: 1, 2, and 3. In addition, the specification of the hardware and software used in this experiment is as follows:

- Operation system: Windows 11
- Processor: AMD Ryzen 5
- Memory: 8GB
- Programming language: Python 3.10
- Libraries: Python3.10, MySQL-connector-Python8.0, numpy1.23, pandas1.5, scikit-learn1.2, Flask2.2, Git2.33
- Database: MySQL
- Web service: Flask

Proposed Method

In this study, several shortest-path algorithms were implemented to find the best cargo route, especially in Indonesia. Greedy, Best First Search (BFS), Dijkstra's, A*, and Floyd-Warshall are implemented, and the performances are investigated based on runtime and validity of results. A more detailed explanation of each algorithm can be seen in the following explanation.

Weighting and Performance Metrics

To find the weights need to do normalization or feature scaling, then calculate the weights with the basic formula as follows:

$$g = \left(price \times \frac{1}{w_1}\right) + \left(distance \times \frac{1}{w_2}\right) + \left(-rating \times \frac{1}{w_3}\right) + (duration \times 1/w\_4) \quad (1)$$

where,
distance: Distance between nodes in kilometers.
price: Travel price between nodes
rating: Review rate of the route
duration: Travel time in hours
g: Actual cost
w: Weight

These parameters are customizable, so users can easily increase or decrease them for more flexible weighting implementation. In this implementation, the

main objective of shortest path algorithms is to find the best path according to the lowest cost value. Distance, price, and duration are parameters that affect the higher cost. Therefore, a lower value means better. For example, in the real world, people will prefer routes with shorter distances, lower prices, and lower duration. Hence, the higher value of w1, w2, and w4 will make them lower. In contrast, a high rating value means a favorite route or cargo service. The high value of w3 can emphasize the importance of rating parameters.

In this study, two parameters to measure the performance of algorithms are runtime and validity. The runtime of an algorithm is a critical factor to consider when solving the shortest path problem in large-scale graphs, as it determines the efficiency and practicality of the solution. The runtime of a shortest path algorithm is typically measured in terms of the number of operations it performs, such as the number of comparisons, assignments and arithmetic operations. The runtime of a shortest path algorithm depends on several factors, such as the size of the graph, the complexity of the algorithm, and the specific implementation of the algorithm. The validity of a shortest path algorithm is determined by its ability to accurately and consistently find the shortest path between two nodes in a given graph. The shortest path algorithm is valid if it always returns the shortest (best) path from a source node to a destination node, given the weights of the edges in the graph.

Experimental Setup

In this experiment, the process is divided into several steps such as data collection, building graph data structure, calculating cost, searching the best route, and performance evaluation as shown in Fig. 1.

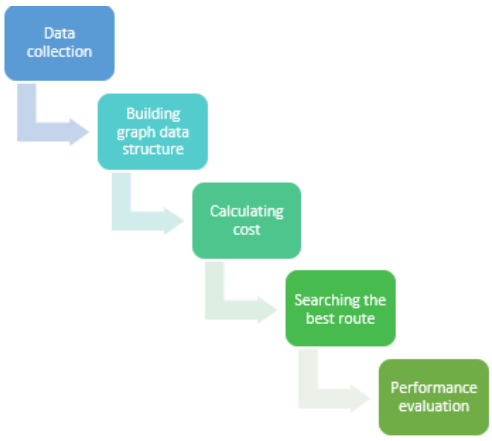

Fig. 1: Process of experiment

Greedy Algorithm

The Greedy algorithm is one for solving optimization problems. The optimization problem in question is the problem of finding the optimum solution, namely maximization or minimization. The Greedy algorithm finds the shortest path by selecting the locally best step at each iteration. At each step, many choices need to be evaluated so that this algorithm will choose the best step (Azis *et al*., 2018a):

- Getting the best choice (local optimum)
- Unable to change a decision or return to the previous step
- The selection of local optimum must result in global optimum

The greedy algorithm operates under the assumption that selecting the locally optimal choice at each step will lead to a globally optimal solution. There are the following six elements of the Greedy algorithm (Azis *et al*., 2018a):

Candidate the Association (C): The set contains the elemental solutions of the various candidates that will take the next step.

Set of Solution (S): The set consists of a group of candidate solutions that have been selected as potential solutions to a problem, and it forms an integral part of the candidate's pool.

Selection Function: This function is to select the candidate who has a high chance of getting the optimal solution. Candidates who have already been selected will not be reconsidered at the next step.

Feasible Function: The purpose of this function is to verify that the selected candidate is capable of providing a suitable solution, thereby preventing any violation of the constraints in the candidate and solution set. Candidates that satisfy the criteria will be included in the solution set, while those that fail to meet the specified requirements will be rejected.

Objective Function: This function changes the value of the solution to the maximum or minimum. The following steps outline the process of utilizing the Greedy algorithm to determine the shortest route:

- Begin by verifying that the current node is the starting node. If it is, proceed to step (3), but if it is not, then proceed to step (2)
- Next, verify whether the destination node is the most recently visited node. If it is, proceed to step (3), but if it is not, then go to step (3)
- Check to determine which side is connected to the next node as well as the current node
- To go to the next node (local optimum), find the side that has the lowest weight

- After according to the local optimum that has been determined, proceed to the next node
- So that the desired destination node can be found, go back to step (1)

The pseudo-code of the BFS algorithm can be seen as follows:

Algorithm 1 (Januantoro *et al*., 2021): Pseudo-code Greedy algorithm

```
Declaration:
    int J[n]
local:
     int result // shortest route distance
result ← 0
for x ← 1 to n do:
    if (result < J[x]) then
        result ← J[x]
    endif
end
```

Best First Search (BFS) Algorithm

BFS is a shortest path algorithm using heuristic values where node expansion depends on the evaluation function *f(v).* This function estimates how far the distance is between node *v* and the destination node by utilizing the heuristic function *h(v) (ie f (v) = h(v)).* A node that has a lower value has a greater chance of leading to the destination node. In general, estimating *f(v)* depends on information about node v and the destination node. Information that has a significant effect and depends on *f(v)* is information taken from the problem domain (Khattab *et al*., 2022). The BFS algorithm also aims to find the shortest path by prioritizing the steps that are closest to the destination. This makes BFS a suitable algorithm for the comparison shortest path in finding cargo routes.

The pseudo-code of the BFS algorithm can be seen as follows:

Algorithm 2 (GBFSF, 2023): Pseudo-code BFS algorithm

```
procedure GreedyBFS (startNode, targetNode):
    mark startNode as visited
    add startNode to nodeQueue
    while nodeQueue is not empty do:
        currentNode ← vertex of nodeQueue with min distance to targetNode
        remove currentNode from nodeQueue
        foreach neighborNode n of currentNode do:
            if n not in visited then:
                if n is targetNode:
                    return n
                else:
                    mark n as visited
                    add n to nodeQueue
    return failure
```

Dijkstra's Algorithm

Dijkstra's is one of the shortest path algorithms that is used to find the optimal path length between two nodes in a graph. Optimal here can mean distance, cost, time, and others. Below includes an explanation of how to step by step Dijkstra's algorithm (Thareja, 2018):

1) Choose a starting point, also known as the A* node, from which the A* algorithm will begin its search
2) Create an empty set N that will be used to store the vertices with the shortest paths that have been discovered
3) Label the starting node and add it to set N
4) If the destination node has been entered or there are no more nodes labeled N, then repeat steps 5 to 7
5) Consider each vertex connected by the edge of the new vertex and the added vertex is not in N
6) (a) If a node not in set N does not have a label, then assign the label of the newly inserted node plus the length of the edge to that node. (b) Otherwise, if a node not in set N already has a label, then set its label to the minimum value between the sum of the newly inserted node's label and the edge length, and the old label
7) Choose a node that is not in set N and has the smallest assigned label, and then include it in set N

Dijkstra considers the weights of the edges, resulting in accurate shortest path calculations. In Dijkstra's algorithm, every vertex in the graph is assigned a label, which can either be temporary or permanent. Nodes that have not yet been visited are assigned temporary labels in Dijkstra's algorithm. In contrast, visited nodes whose distance (weight) is known are given the label details. A node in Dijkstra's algorithm can only be assigned either a permanent or a temporary label, but not both. The algorithm is executed as follows (Thareja, 2018):

1) In Dijkstra's algorithm, the destination node is assigned a label that represents the shortest distance from the source node to the destination node
2) In Dijkstra's algorithm, if the destination node does not have a label assigned to it, then it indicates that there is no shortest path from the source node to the destination node

The pseudo-code of Dijkstra's algorithm can be seen as follows:

Algorithm 3 (Mehlhorn and Sanders, 2008): Pseudo-code Dijkstra's algorithm

```
1  function Dijkstra's (Maps, origin):
2      foreach vertex i in Maps.Vertices:
3          dist[i] ← INFINITY
4          prev[i] ← UNDEFINED
5          add i to G
6      dist[source] ← 0
7      while G is not empty:
8          u ← vertex in G with min dist[u]
9          remove u from G
10         foreach neighbor v of u still in G:
11                 alt ← dist[u] + Graph.Edges(u, v)
12                 if alt < dist[i]:
13                     dist[i] ← alt
14                     prev[i] ← u
15     return dist[], prev[]
```

A Algorithm*

A* algorithm is a popular and effective method that is often recommended as a solution to the problem of finding optimal paths between nodes in a graph or network. The A* algorithm is designed to search for the most efficient or shortest path between a starting node and a destination node in a graph or network. This simulation employs the A* algorithm with a heuristic function as its search technique (Budiman *et al*., 2018). By utilizing a suitable heuristic function, A* efficiently finds the shortest path.

Applying the shortest path, A* implies the maintenance of two lists, namely an OPEN list and a CLOSED list. OPEN contains nodes apprised by the heuristic function but has not yet been extended to its successors, whereas CLOSED contains nodes that have been visited. The A* shortest path algorithm is one of the best algorithms but does not always produce the shortest path because it relies on heuristics (Budiman *et al*., 2018).

By combining the advantages of uniform cost search and Greedy search, the A* algorithm is regarded as the Best First Search (BFS) algorithm. The considered price is obtained from the actual price plus the estimated price. In mathematical notation, it is written:

$$f'(n) = g(n) + h'(n) \tag{2}$$

(by using $f'(n)$, the A* algorithm is complete and optimal)

where,

$f'(n)$: lowest estimated cost
$g(n)$: cost from the initial node to node
$h'(n)$: estimated cost from node n to the final node

Some of the functions of A* are as follows:

1. The A* algorithm maintains a tree structure that represents all possible paths originating from the starting node
2. Extend the path one side at a time
3. Continue until the termination criteria are listed

The pseudo-code of the A* algorithm can be seen as follows:

Algorithm 4 (Hadi Nuryoso, 2020): Pseudo-code A* algorithm

```
1  Function A* (start, destination)
2  closedset ← empty sets
3  openset ← {start}
4  origin← empty map
5  q_value [start] := 0;
6  f_value ← q_value + estimation_value_heuristic
   (start, destination)
7  While openset is not empty:
8      current ← node in lowest openset f_value[]
9      if current ← destination
10     Return path_construction
       (Origin, destination)
11     Remove current from openset
12     Add current to closedset
13     Foreach inner neighbor node_neighbor(current):
14     if neighbor is inside closedset:
15     Continue q_temporary_value ←
       q_value[current_node]+
       dist_between(current_node)

16     if the neighbors not in openset:
17        q_value_temporary < ← q_value[neighbor]
18           origin[neighbor] ← current
19           q_value[neighbor] ← q_temporary_value
```

Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is a classic algorithm for solving the all-pairs shortest path problem, which computes the shortest path between every pair of vertices in a weighted graph (Azis *et al*., 2018b). The way this algorithm solves the problem is to choose a solution based on the previous solution with conditions that must be related to each other, then it will produce a lot of solutions (Azis *et al*., 2018a). The Floyd-Warshall algorithm is a straightforward and efficient algorithm used to find the shortest path between all pairs of nodes in a graph. The Floyd-Warshall algorithm takes as input a weighted graph that is directed, and this graph is usually represented as a list of edges and vertices. The Floyd-Warshall algorithm computes the minimum distance between all pairs of vertices in a weighted graph by considering all intermediate vertices along the way, until the final shortest

distance between any two pairs of vertices is obtained (Azis *et al*., 2018a). The Floyd-Warshall algorithm compares all possible paths between every pair of vertices in a graph and selects the shortest path at each stage, resulting in an optimal estimate for the shortest path between each pair of vertices. However, the processing time can be slow and may not be suitable for cases with large data due to the long graph initialization process. Although it has a higher time complexity compared to other algorithms, the Floyd-Warshall algorithm is still a popular choice for calculating the shortest path due to its ease of implementation and practicality. It can efficiently determine the shortest paths, albeit with slightly higher time complexity compared to other algorithms:

The pseudo-code of the Floyd-Warshall algorithm can be seen as follows:

Algorithm 5 (Algorithms Notes for Professionals, n.d.): Pseudo-code Floyd-Warshall algorithm

```
1   Let dist be a |X| × |X| array of minimum distances initialized to ∞ (infinity)
2   foreach edge (x, y) do:
3       dist[x][y] ← w (x, y)
4   foreach vertex v do:
5       dist[y][y] ← 0
6   for u from 1 to |X|:
7       for v from 1 to |X|:
8           for z from 1 to |X|:
9               if dist[v][z] > dist[v][u] + dist[k]
10                  dist[v][z] ← dist[v][u] + dist[u][z]
11              endif
12          end for
13      end for
14  end for
```

Results and Discussion

In this section, the experimental setup is discussed, and the results based on the utilization of five algorithms are presented. A cargo route optimization application was developed using the following algorithms: Greedy, Dijkstra's, Best First Search (BFS), A*, and Floyd-Warshall. Table 1 provides a comparison of the shortest path algorithms. The execution time of each algorithm was measured in milliseconds. In Table 1, the execution times of the algorithms can be directly compared, indicating which algorithm performs the fastest. If there is a significant difference in execution time between two algorithms, it can be concluded that one algorithm is more efficient than the other.

The BFS algorithm demonstrated the best performance with the shortest average execution time. The BFS algorithm achieves the shortest runtime by performing a horizontal search on the levels of nodes in the graph. It starts by examining all nodes connected to the starting node, then checks the nodes connected to those nodes on the next level, and continues until it reaches the target node. Since this algorithm examines all nodes at each level, BFS can quickly find the shortest path on a less complex graph. Interestingly, some of the shortest execution times resulted in false recommendation routes. Despite having the shortest execution time, search algorithms have the potential to produce inaccurate routes. Finding the shortest execution time solution does not always mean finding the best solution. This can be due to a large search space or the algorithm not being suitable for the specific case study. In such cases, the algorithm may not be able to explore all possible paths. Table 1 also shows that the Greedy algorithm has the longest execution time. This is because the Greedy algorithm makes choices based solely on the information available at the current step of the search. It selects the shortest path to the next intersection without considering other factors such as traffic conditions, road capacity, or potential congestion. This can result in suboptimal or even inappropriate routes that take longer to traverse than other routes. Additionally, the Greedy algorithm generates false recommendation routes, with some results producing looping routes. Based on the weight optimization and dataset used, it can be concluded that the BFS and Greedy algorithms are not the most effective options.

The Dijkstra's, A*, and Floyd-Warshall algorithms are compared. These algorithms can be more effective in finding the shortest or best path and still have reasonable execution times. Interestingly, the A* algorithm achieved the shortest execution time while providing better recommendation routes compared to the other three algorithms. The A* algorithm is a heuristic search algorithm that uses a heuristic function to guide the search towards the goal state, resulting in a more efficient search for the shortest path. The heuristic function estimates the distance from the current node to the goal node and uses this estimate to prioritize the search towards the goal. In contrast, other algorithms such as Dijkstra's algorithm do not use a heuristic function and consider all possible paths from the starting node to the goal node, which can be computationally expensive. Therefore, the A* algorithm often finds the shortest path more quickly, especially in larger graphs. Additionally, the recommendation routes provided by the A* algorithm may be more suitable because the heuristic function considers factors such as distance, traffic, or other constraints that can affect the actual travel time.

Table 1: The optimization results from five algorithms. ASP means according to the shortest path. N-ASP means the path does not use the shortest path

Algorithm	Hops	Status	Runtime (ms)	Average runtime (ms)	Description
Greedy	1	ASP	9,089	11,072.5	-Some results do not match the route
	2	ASP	8,574		and there is a possible route not found
	2	ASP	9,224		or looping forever
	3	N-ASP	17,403		-Longest average runtime
BFS	1	N-ASP	6,959	6,734.67	-Some results do not match the route
	2	ASP	6,288		-Shortest average runtime
	3	N-ASP	6,957		
Dijkstra's	1	ASP	6,986	7,404.34	-All recommendations are the best route
	2	ASP	8,106		-Longer average runtime than A*star
	3	ASP	7,121		
A*	1	ASP	6,049	7,056.67	-All recommendations are the best route
	2	ASP	8,067		-Shortest average runtime
	3	ASP	7,054		
Floyd-Warshall	1	ASP	10,971	8,978.67	-All recommendations are the best route
	2	ASP	8,363		-Longer average runtime than A*star
	3	ASP	7,602		

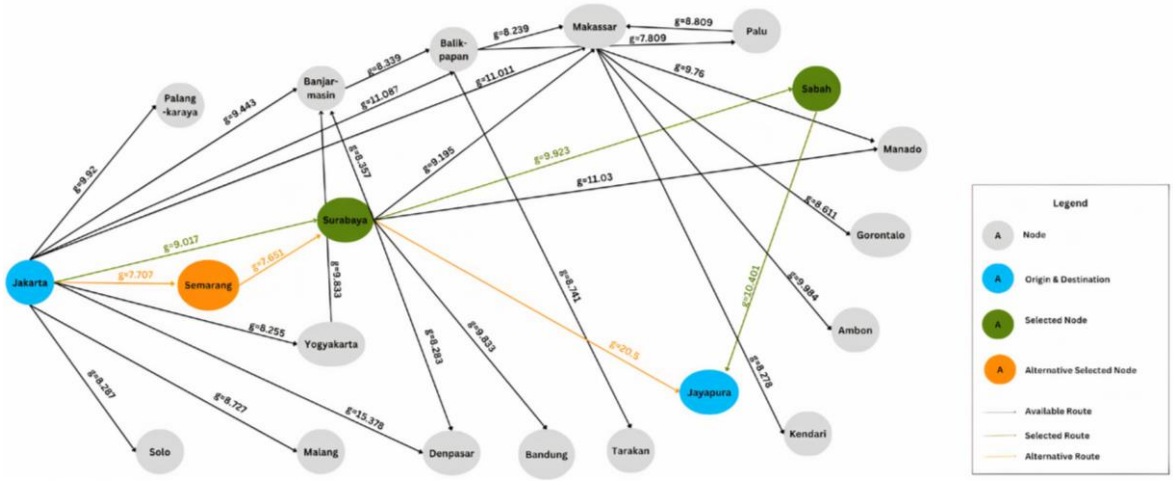

Fig. 2: Visualization map of A* algorithm

The utilization of Dijkstra's, A*, and Floyd-Warshall algorithms resulted in the shortest paths, consistent with prior research findings (Mehlhorn and Sanders, 2008). This supports previous studies that have shown the effectiveness of these algorithms in determining the shortest routes in similar problem contexts. However, it is worth noting that the A* algorithm demonstrated better performance in terms of execution time compared to the other algorithms, making it a more efficient choice for determining the shortest path.

Furthermore, it was found that the Greedy algorithm generated looping or incorrect routes. This finding aligns with previous research highlighting the limitations of the Greedy algorithm in finding the best routes.

Overall, this research confirms previous findings and demonstrates that the use of algorithms such as Dijkstra's, A*, and Floyd-Warshall can provide more optimal solutions for finding the shortest paths in this research context. These results are consistent with existing literature and do not present significant novelty. However, it is important to consider that the specific implementation and context of this research contribute to the advancement of knowledge in this field.

Figure 2 shows an example of a visualization map. The line is a directional connector between nodes. While numbers are weights calculated from the algorithms. The shortest route recommendations are represented with green nodes. In addition, g is actual cost used in Eq. (1). For example, this scenario is to find the best cargo route from Jakarta to Jayapura. According to A* algorithm, the best route is Jakarta-Surabaya-Sabah-Jayapura. Testing was performed on the aforementioned algorithms utilizing a self-created dataset obtained through web scraping from the browser. Among them, there is a dataset of cities that contains columns for id, city name, latitude, and longitude that will be used in the sample table. Furthermore, a

sample dataset was available, comprising columns for id, origin, and destination, which included city IDs from the city table. It also featured several parameter columns for storing data related to distance, price, rate, and duration. The more detailed dataset can be downloaded in this link: https://github.com/JstKhalid/DoRoute/blob/main/dataset_cargo_route.xlsx.

Conclusion

The novelty of this research lies in the investigation and comparison of five different shortest path algorithms (Greedy, Dijkstra's, Best First Search, A*, and Floyd-Warshall) for cargo route optimization. While previous studies may have focused on one or two algorithms, our research expands the scope by considering multiple algorithms and evaluating their performance in terms of runtime and route validity.

Furthermore, the inclusion of the Greedy algorithm in the evaluation is a unique aspect of our study. This algorithm, although widely used in various applications, is known to have limitations in terms of finding the best route and potential issues such as looping or failure to find a route. By incorporating the Greedy algorithm into the analysis, the study sheds light on its limitations and presents a comprehensive comparison with alternative algorithms.

Additionally, the evaluation is conducted in different test scenarios with 1, 2, and 3 hops. This allows us to assess the performance of the algorithms under varying network complexities and distances. The results indicate that while some algorithms may perform well in terms of runtime, it may not always yield the best route. Conversely, algorithms such as Dijkstra's, A*, and Floyd-Warshall consistently provide the optimal route, with A* standing out as the fastest among them.

In conclusion, our research contributes to the field of cargo route optimization by providing a thorough investigation and comparison of multiple shortest path algorithms. The inclusion of the Greedy algorithm and the evaluation in different scenarios add novelty to the study. The findings highlight the strengths and weaknesses of each algorithm, ultimately concluding that A* is the most suitable algorithm for cargo route optimization.

With this research, some of the problems that can be addressed by the developed and compared model include:

1. Route optimization: The model can assist in optimizing the routes for cargo delivery or transportation by finding the shortest or best paths that minimize time, distance, cost, and ratings
2. Logistics efficiency: The model can help improve efficiency in the supply chain and logistics processes by enhancing delivery processes and optimizing resource utilization
3. Schedule planning: The model can be used to plan delivery or distribution schedules by considering optimal delivery times
4. Strategic decision-making: The model can provide essential information for strategic decision-making in logistics

By addressing these problems, the proposed model offers potential solutions to enhance various aspects of logistics operations.

Limitations and Future Research

This research only implements 5 traditional algorithms. Therefore, for future research, it may be possible to implement more than 5 search algorithms.

Acknowledgment

Authors would like to thank Telkom University for its research funding.

Funding Information

This research is funded by Telkom University.

Author's Contributions

Dedy Rahman Wijaya: Methodology, project administration, investigation, validation, conceptualization, formal analysis, written reviewed and edited, supervision, funded acquisition, resources.

Aqil Athaliah: Software, data curation, validation, investigation, written original drafted visualization.

Tiara Nuha Noor'afina: Written original drafted, software, validation, investigation, data curation, visualization.

Patrick Adolf Telnoni: Methodology, investigation.

Sari Dewi Budiwati: Written, reviewed and edited.

Ethics

The corresponding author affirms that the manuscript is original and contains unpublished material and that all co-authors have reviewed and approved it. The authors also confirm that the research adheres to ethical principles and guidelines, and that no ethical issues are involved in the study. In addition, all authors should have contributed significantly to the research and agreed to the publication of the manuscript.

References

Algorithms Notes for Professionals. (n.d.). goalkicker.com

Amin, C., Mulyati, H., Anggraini, E., & Kusumastanto, T. (2021). Impact of maritime logistics on archipelagic economic development in eastern Indonesia. *Asian Journal of Shipping and Logistics*, *37*(2), 157-164. https://doi.org/10.1016/j.ajsl.2021.01.004

Azis, H., Lantara, D., & Salim, Y. (2018a, November). Comparison of Floyd-Warshall algorithm and greedy algorithm in determining the shortest route. In *2018 2nd East Indonesia conference on computer and information technology (EIConCIT)* (pp. 294-298). IEEE. https://doi.org/10.1109/EIConCIT.2018.8878582

Azis, H., Mallongi, R. dg., Lantara, D., & Salim, Y. (2018b). 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT). IEEE. ISBN: 10-1538680505; 9781538680506.

Budiman, V., Agung, H., & Leksmono, Y. S. H. (2018). Aplikasi Berbasis Android Untuk Mencari Lokasi Puskesmas Terdekat Dengan Algoritma a-Star Di Provinsi Dki Jakarta. *JUST IT: Jurnal Sistem Informasi, Teknologi Informasi dan Komputer*, *9*(1), 39-48. https://doi.org/10.24853/justit.9.1.39-48

Cheng, D., Gkountouna, O., Züfle, A., Pfoser, D., & Wenk, C. (2019, November). Shortest-path diversification through network penalization: A washington DC area case study. In *Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science* (pp. 1-10). https://doi.org/10.1145/3357000.3366137

Evangelista, D. G. D., Vicerra, R. R. P., & Bandala, A. A. (2020, December). Approximate Optimization Model on Routing Sequence of Cargo Truck Operations through Manila Truck Routes using Genetic Algorithm. In *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)* (pp. 1-5). IEEE. https://doi.org/10.1109/HNICEM51456.2020.9400044

Gbadamosi, O. A., & Aremu, D. R. (2020, March). Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICMCECS47690.2020.240873

GBFSF. (2023). Greedy Best-First Search when EHC Fails. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume28/coles07a-html/node11.html#modifiedbestfs

Hadi Nuryoso, P. Y. (2020). Penerapan Algoritma A* pada Pencarian Rute Terpendek pada Rute Angkot Di Kota Sukabumi. *Jurnal Sarjana Teknik Informatika*, *8*(1), 21-35.

Ireland, S. M., & Martin, A. C. R. (2020). Atomium-a Python structure parser. *Bioinformatics*, *36*(9), 2750-2754. https://doi.org/10.1093/bioinformatics/btaa072

Januantoro, A., Septiyanto, A. F., & Hapantenda, A. K. W. (2021). konvergensi_volume_17_nomor_1_januari_20. jurnal konvergensi untag surabaya, 17, 47–55

Ju, C., Luo, Q., & Yan, X. (2020). Path Planning Using an Improved A-star Algorithm. *Proceedings 11th International Conference on Prognostics and System Health Management, PHM-Jinan 2020*, 23-26. https://doi.org/10.1109/PHM-Jinan48558.2020.00012

Khattab, H., Mahafzah, B. A., & Sharieh, A. (2022). A hybrid algorithm based on modified chemical reaction optimization and best-first search algorithm for solving minimum vertex cover problem. *Neural Computing and Applications*, *34*(18), 15513-15541. https://doi.org/10.1007/s00521-022-07262-w

Liana, L. I., & Nudin, S. R. (2020). Implementasi Algoritma Best-First Search untuk Aplikasi Mesin Pencari Handphone pada E-commerce (Apenphone). *Journal of Informatics and Computer Science (JINACS)*, *2*(01), 67-73. https://doi.org/10.26740/jinacs.v2n01.p67-73

Liu, F., Tang, X., & Yang, Z. (2018, November). An encoding algorithm based on the shortest path problem. In *2018 14th International Conference on Computational Intelligence and Security (CIS)* (pp. 35-39). IEEE. https://doi.org/10.1109/CIS2018.2018.00016

Mavakala, A. W., Adoni, W. Y. H., Aoun, N. Ben, Nahhal, T., Krichen, M., Alzahrani, M. Y., & Kalala, F. M. (2023). COV19-Dijkstra: A COVID-19 Propagation Model Based on Dijkstra's Algorithm. *Journal of Computer Science*, *19*(1), 75-86. https://doi.org/10.3844/jcssp.2023.75.86

Mehlhorn, K., & Sanders, P. (2008). Chapter 10. Shortest Paths" (PDF). *Algorithms and Data Structures: The Basic Toolbox*. https://doi.org/10.1007/978-3-540-77978-0

Kargo. (2021). Peranan Penting Transportasi Logistik di Indonesia Kargo. https://kargo.tech/blog/transportasi-logistik-di-indonesia/

Phan, T., & Do, P. (2018, February). Improving the shortest path finding algorithm in apache spark graphx. In *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing* (pp. 67-71). https://doi.org/10.1145/3184066.3184083

Qian, L., & Yinfa, Z. (2018, August). A Shortest Path Algorithm Under Specified Nodes Constraint. In *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)* (pp. 686-689). IEEE. https://doi.org/10.1109/SDPC.2018.8664822

Qiu, Y. X., Wen, D., Qin, L., Li, W., Li, R. H., & Ying, Z. (2022). Efficient shortest path counting on large road networks. *Proceedings of the VLDB Endowment.* https://doi.org/10.14778/3547305.3547315

Raza, E., Sabaruddin, L. O., & Komala, A. L. (2020). Manfaat dan Dampak Digitalisasi Logistik di Era Industri 4.0. *Jurnal Logistik Indonesia*, *4*(1), 49-63. http://ojs.stiami.ac.id

Rizi, F. S., Schloetterer, J., & Granitzer, M. (2018, August). Shortest path distance approximation using deep learning techniques. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 1007-1014). IEEE. https://doi.org/10.1109/ASONAM.2018.8508763

Sao, P., Kannan, R., Gera, P., & Vuduc, R. (2020, February). A supernodal all-pairs shortest path algorithm. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (pp. 250-261). https://doi.org/10.1145/3332466.3374533

Shan, D., Zhou, W., & Wang, J. (2018, July). A novel personalized dynamic route recommendation approach based on pearson similarity coefficient in cooperative vehicle-infrastructure systems. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)* (pp. 1270-1275). IEEE. https://doi.org/10.1109/CYBER.2018.8688301

Thareja, R. (2018). Data Structures Using C (2nd Ed.). Oxford University Press. pp: 560. ISBN: 10-0198099304.

Wang, J., Wu, N., & Zhao, W. X. (2021). Personalized route recommendation with neural network enhanced search algorithm. *IEEE Transactions on Knowledge and Data Engineering*, *34*(12), 5910-5924. https://doi.org/10.1109/TKDE.2021.3068479

Widiyanto, P., & Nashrullah, N. (2020). The role of transportation and logistics infrastructure in increasing MSMEs in Indonesia (study in the new normal era). *Sustainable Competitive Advantage (SCA)*, *10*(1), 558-567. http://www.jp.feb.unsoed.ac.id/index.php/sca-1/article/view/1918

Wijaya, D. R. (2023). DoRoute (1.0). Zenodo. https://zenodo.org/records/7677821

Wongso, R., Cin, C., & Suhartono, J. (2018). TransTrip: A Shortest Path Finding Application for Jakarta Public Transportation using Dijkstra Algorithm. *J. Comput. Sci.*, *14*(7), 939-944. https://doi.org/10.3844/jcssp.2018.939.944