Available online at :
http://ejournal.amikompurwokerto.ac.id/index.php/telematika/

Telematika

Accreditated SINTA "2" Kemenristek/BRIN, No. 85/M/KPT/2020

Effectiveness of Pickup and Delivery Services in Logistics Companies with Route Optimization using the A* Algorithm

Cahyo Prianto[1], Nur Tri Ramadhanti Adiningrum[2]

[1,2] Applied Bachelor of Informatics Engineering
[1,2] Universitas Logistik dan Bisnis Internasional, Bandung, Indonesia
E-mail: cahyo@ulbi.ac.id[1], 1204061@std.ulbi.ac.id[2]

ARTICLE INFO

History of the article:
Received January 25, 2024
Revised July 24, 2024
Accepted August 6, 2024

Keywords:
A* Algorithm,
Effectiveness,
Pickup,
Delivery,
Route Optimization.

Correspondece:
E-mail: cahyo@ulbi.ac.id

ABSTRACT

Logistics is situated at the epicenter of both production and consumption, its role in the economy is becoming more and more significant. A logistics company is a business that specializes in offering logistics services; an example of such a business in Bandung is a logistics company that offers pickup and delivery services. Of the many locations that will be visited by couriers every day, of course, effective vehicle route management is needed to minimize costs, time, and vehicle efficiency. Therefore, the goal is to find the shortest route from one location to another based on the distance factor. To achieve this goal, the A* algorithm is used using Python as a solution to find the shortest route and Dijkstra as a comparison of route search algorithms. The study's findings demonstrated that the A* algorithm, with a calculation time of 0.0004022 ms, was efficient in finding the shortest path while requiring the least amount of CPU processing at 5.56%. While Dijkstra took 7.29% of the computation and produced a time of 0.033026 ms. A* proved effective in finding the shortest route by producing a distance of 3.11 km. While other routes produced distances of 3.34 km, 4.54 km, and 4.77 km. In addition, the use of a GUI has been successfully implemented as an interactive visualization so that couriers can easily find the shortest route along with the distance traveled. The logistics company can use the A* algorithm and the GUI developed to improve the efficiency of delivery and pickup of goods. By utilizing optimized shortest route searches, companies can save time and increase customer satisfaction through faster and more efficient delivery.

INTRODUCTION

Because logistics is situated at the hub of both production and consumption, its significance in a market economy is growing(Zhu & Zhu, 2023). Logistics management is crucial to the supply chain because it manages the smooth and efficient movement of funds, information, and items between businesses in order to meet the needs of final customers(Cano et al., 2021). In order to accomplish this, logistics management needs to leverage technology to support its business activities, including service management, transportation, distribution, customer service, and other operations, that effectively captures, processes, and transmits information(Winkelhaus & H. Grosse, 2019).

Transportation, distribution, and warehousing are just a few of the supply chain operations that logistics companies assist their clients with managing(Ngo, 2023). One of them is a logistics company in the city of Bandung that provides pickup and delivery services. In this service, the activity planning and time management stages are the stages that determine the success of a pickup and delivery service. The

following are the main obstacles to last-mile delivery: cost, time constraints, sustainability, growing volumes, and an aging workforce(Boysen. Nils et al., 2020). For instance, delivery trucks drive erratically and go needless distances(Jiang & Mahmassani, 2014) and causes expenditures of up to 60% of total emissions (Dablanc et al., 2011).

Based on this, the need for optimal vehicle routes for pickup and delivery services becomes important (Pane et al., 2019) in pickup and delivery services. This is because in this modern era, the logistics system needs to undergo a transformation in order to produce a more efficient model, with as little negative impact as possible on the environment(Palacin et al., 2023). Therefore, logistics blended with technology has become a promising solution to deal with such problems(Feng & Ye, 2021).

Even though there are various route search algorithms available, their application in the context of pickup and delivery services in logistics companies in the city of Bandung is still flexible. A research gap arises in the lack of specific studies regarding the application of shortest route search algorithms in this context. Therefore, this research is recommended to fill this knowledge gap by focusing on finding the shortest route for pickup and delivery services in the specific context of logistics companies in the city of Bandung.

Numerous graph search algorithms are available in the scientific literature that may search a graph or navigation tree to determine the shortest path between two nodes(Palacin et al., 2023). The pathfinding research community has long employed the search algorithm A*. Its effectiveness, ease of use, and adaptability are frequently cited as advantages over other instruments. A* has become a popular choice among academics for solving pathfinding problems due to its versatility and widespread use(Foead et al., 2021). Since no other optimization algorithm can guarantee the expansion of fewer nodes, A-star search has the primary benefit and is widely used to explore thick or enormous graphs in challenging issues(Foead et al., 2021).

Thus, the goal of this study is to determine the most practical path—in this example, the shortest path given the distance factor—for logistics firms' pickup and delivery services. Put differently, it searches through all of the current routes for the shortest path that will get it from the starting point to the destination with the least amount of weight(Gede, 2018). Apart from that, effective routes are also created to avoid back and forth work being carried out by couriers. In addition, stopping locations can be applied if the courier will visit more than one location. So the desired location can be used as a route consideration so that couriers can also visit these locations.

This research will focus on logistics companies based in the city of Bandung, Indonesia, but with their identities disguised to maintain confidentiality. The data used is the main branch office as the starting point and ending point along with the addresses and pickup and delivery routes in the form of latitude and longitude from the location point. To achieve the research objectives, the A* Algorithm with the Python programming language was used as the calculation process used. To perform pickup and delivery service activities, the A* algorithm finds the shortest path from the main branch office to customer addresses. In this research, the use of the A* algorithm is limited to considering distance as a determining factor for finding the shortest path. Thus, the variable considered in this study is only distance, with other factors such as time, cost, or traffic conditions not considered, with the assumption that other factors are insignificant or can be ignored in the context of shortest path search. The discussion will be significantly strengthened by including a comparative analysis with other pathfinding algorithms, namely the Dijkstra algorithm. This comparison will include the resulting route distance, computation time, and CPU usage. A graph's shortest

path can be found using the Dijkstra algorithm(Behún et al., 2022). This will offer a more thorough assessment of the pathfinding algorithm. Thus, a more complete picture of the performance of the algorithm used will be obtained.

The output of the research is to find out the closest vehicle route to carry out pickup and delivery services. It is anticipated that the study's findings will help businesses in the logistics industry, particularly those offering pickup and delivery services, and will also contribute to enhancing those services' effectiveness.

RESEARCH METHODS

Researchers must be familiar with both methodology and research methods/techniques (Patel & Patel, 2019). To accomplish the ultimate research goal, this study was conducted in phases. Research begins with the planning stage as the beginning of determining the topic and determining the research objectives. Then the stage continues for data collection and exploration of supporting theories. The next stage is the stage to analyze the collected data. Then the next step is to implement to create graphs, find heuristic values, and apply A* and Dijkstra. Then for the last stage is to evaluate the work with a comparison of algorithms and create a GUI. The methodology used to achieve the objectives in Figure 1.

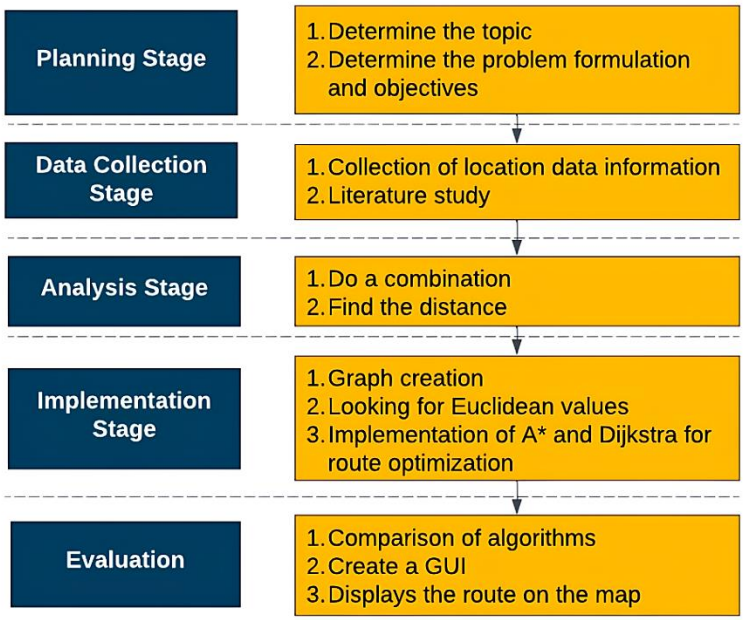

Figure 1. Research Methodology Flow

1. **Planning Stage**

At this stage there are two activities, which are described as follows.

a. Determine the topic

The act of choosing a specific issue or problem to be looked into further with the intention of identifying, evaluating, and comprehending certain parts of the topic in line with the specified study objectives is known as picking a research topic. Where the topic chosen in this research is optimizing pickup and delivery service routes.

b. Determine the formulation of problems and objectives

The issue that needs to be addressed is the requirement for the best possible vehicle route for delivery and pickup services—in this example, the quickest path given the distance factor. To get closer to the trip is the aim. This is due to the fact that route optimization offers advantages

including increasing worker productivity, cutting down on maintenance expenses, and drastically lowering fuel usage(Durdu & Kaya, 2023).

2. Data Collection Stage

At this stage there are two activities, which are described as follows.

a. Collection of location data information

The goal of the data collection method is to obtain understanding of the research issue(Taherdoost, 2021). The practice of gathering facts, information, and notes on historical events or circumstances for use in analysis, study, or a deeper comprehension of how a specific issue or setting developed from the viewpoint of the past is referred to as location data information collection. In this case, the information obtained is the location of the main branch office along with the location of the consumer's address and the location of the branch route. The data obtained are the location coordinates. These location points are called nodes. The node addresses and labels used are as in Table 1.

Table 1. Location Data and Coordinates

Location	*Label*	*Coordinate*	*Location*	*Label*	*Coordinate*
Office	A	(-6.919792, 107.606014)	Location13	N	(-6.925884, 107.607131)
Location1	B	(-6.915593, 107.601029)	Location14	O	(-6.927419, 107.607004)
Location2	C	(-6.927422, 107.610256)	Location15	P	(-6.928340, 107.609310)
Location3	D	(-6.924132, 107.606715)	Location16	Q	(-6.927377, 107.605908)
Location4	E	(-6.916932, 107.604728)	Location17	R	(-6.927251, 107.603658)
Location5	F	(-6.927167, 107.606442)	Location18	S	(-6.920806, 107.604099)
Location6	G	(-6.920149, 107.606520)	Location19	T	(-6.920128, 107.598304)
Location7	H	(-6.921020, 107.606448)	Location20	U	(-6.916292, 107.598179)
Location8	I	(-6.921194, 107.607428)	Location21	V	(-6.916175, 107.600292)
Location9	J	(-6.922484, 107.607500)	Location22	W	(-6.914935, 107.600452)
Location10	K	(-6.922474, 107.607067)	Location23	X	(-6.916161, 107.600585)
Location11	L	(-6.922382, 107.606355)	Location24	Y	(-6.915815, 107.604469)
Location12	M	(-6.925660, 107.606076)	Location25	Z	(-6.915489, 107.606512)

Node A is specified as the starting point, so the route starts from node A and ends at the final destination node according to the request. Information is searched using Google Maps to present a route formed as in Figure 2.

Figure 2. Route Graph with A-Z Labels Based on Google Maps

All node information, both labels and point coordinates, are made into a dataframe in Python as in Table. 2, which consists of the top 5 data out of 26 data.

Table 2. Location Dataframe and Coordinates

Nodes	*Latitude_Node*	*Longitude_Node*
A	-6.919792	107.606014
B	-6.915593	107.601029
C	-6.927422	107.610256
D	-6.924132	107.606715
E	-6.916932	107.604728

b. Literature Review

A literature review is, in general, a methodical approach to gathering and summarizing prior information. A solid basis for knowledge advancement and the facilitation of theory formation is created by an efficient and well-conducted review as a research method(Snyder, 2019). To do this, a number of journals, research papers, or other documents pertaining to or relating to ongoing research are examined(Trivaika & Senubekti, 2022). The study in question is to find the theoretical basis of the A* and Dijkstra algorithms and how to apply them.

The pathfinding research community has long employed the search algorithm A*. The fundamental idea behind A-Star is to repeatedly investigate the least-traveled areas. If the objective is location, then this method is finished. In the event that it is not the intended outcome, A-Star will identify nearby venues and investigate alternative options. The terms starting point, node, open list, closed list, price (cost), and resistance are fundamental to the A-Star algorithm(Candra et al., 2021).

One of the simplest methods for figuring out the slope of a node plot in a given graph is the A-Star (A*) algorithm. In a two-way weighted network, the A* heuristic algorithm is used to determine the shortest path between two vertices(Aswandi & Marlina, 2023). A graph is a structure composed of subunits that multiply an object and are connected by edges that describe the relationship between the object and the subunits(Pranoto, 2020).

Graph calculations use the following formula (1):

$$G = (V, E) \quad (1)$$

Where:

G = Graph

V = A non-empty set contains nodes {v1, v2, v3, ... , vn}

E = A non-empty set contains edges {e1,e2, ..., en}

The A-Star (A*) algorithm is similar to the Djikstra algorithm, but it uses functions called heuristics to estimate the data in the process(Pranoto, 2020). Using heuristic methods to estimate remaining costs from the starting node to the destination point. This condition gives the A star algorithm an opportunity to select the next closest node and optimize the search for the shortest route. The A-Star algorithm is a heuristic search algorithm for finding the optimal path in a static obstacle environment(Sheng et al., 2018). Use of heuristic methods to estimate remaining costs from the starting node to the destination point. This condition provides an opportunity for the A

star algorithm to select the next closest node and optimize the search for the shortest route. Calculation of the shortest route in the A-Star algorithm uses the following formula (2):

$$f(n) = g(n) + h(n) \quad (2)$$

Where:

g = Distance required to move from the starting point to a point n on a map

h = Estimated distance required to move from point n on the map to the destination point (heuristic)

n = distance of a node

(n) = lowest estimated cost

(n) = cost from the initial node to node n

$h(n)$ = estimated cost from node n to the final node

The classic A-star algorithm begins at the starting point, determines the cost of moving the current node to the beginning and finishing points within the evaluation function's limitations, and then expands radially to the target point. This is how the A* algorithm operates. It keeps pathfinding until it reaches the goal point, returning to the area around the beginning point when it comes across an obstacle(Liu et al., 2022).

The A* algorithm's benefit is that it shortens path planning time by eliminating computation and data redundancy. The advantages of the enhanced A-star algorithm will become more clear as the environment gets more complicated(Liu et al., 2022). A-Star's benefit is in its utilization of a heuristic function for optimization, whereby each node is assigned a value that directs A-Star towards achieving the intended outcome(Candra et al., 2021). Its effectiveness, ease of use, and adaptability are frequently cited as advantages over other instruments. Owing to its extensive and pervasive application, A* has emerged as a popular option among researchers attempting to resolve pathfinding issues(Foead et al., 2021). A*'s drawback, aside from its benefits, is that complex data, like the straight line distance to the node (final state), must be included in the graph(Wayahdi et al., 2021).

The Dijkstra algorithm is a method for resolving the issue of determining the shortest path between vertex a and z with the least possible length. The purpose of this algorithm is to determine the quickest route and handle difficulties involving routes to specific locations(Firwanda et al., 2021). The Dijkstra algorithm calculates the shortest path by starting at the origin and going through the nearest point, the second point, and so forth. Finding the cost value that functions closest to the goal in a weighted graph is the fundamental notion behind the Dijkstra algorithm, which helps in path selection. For example, the Dijkstra algorithm determines all the minimum weights from each point, where points represent buildings and lines represent highways. This algorithm's general goal is to determine the shortest path between two points using the least amount of weight (Wulandari & Sukmasetya, 2022).

The advantage of the resolution mechanism in the Dijkstra Algorithm lies in the process, namely the initial process of selecting which point will be the distance weight at the node which in the next development stage searches for the origin from one point to another and to the next

point gradually until finding the final node that is intended as the intended point (Bismi et al., 2021).

3. Analysis Stage

At this stage there are two activities, which are described as follows.

a. Do combinations

At this stage, a combination of all location points, including the main branch office, consumer addresses, and branch route locations (which will later be called nodes) from the 26 existing location points, is carried out. An example of a combination result is where A visits B, then A visits C, then A visits D, or B visits D, and so on.

Table 3. The Code Creates a Combination of Nodes

```
from itertools import combinations
# Take all combinations of 26 nodes
nodes = df['Nodes'].tolist()
possible_paths_combinations = list(combinations(nodes, 2))
# Count the number of combinations
num_possible_paths_combinations = len(possible_paths_combinations)
print(f"The number of combinations of 26 nodes is:
{num_possible_paths_combinations}")
```

Based on Table 3, using the combinations library from itertools can be used to calculate how many combinations are obtained from 26 nodes. The combined results show that there are 325 one-way routes.

b. Find the distance

After carrying out the combination process of all nodes, all possible routes are searched for their respective distances. From Table 4, The Haversine formula is used to compute the distance (in kilometers) between two geographic coordinate points using the haversine function.

Table 4. The Code Calculates the Distance between Nodes

```
def haversine(lat1, lon1, lat2, lon2):
    R = 6371.0  # Earth radius in kilometers
    lat1_rad = np.radians(lat1)
    lon1_rad = np.radians(lon1)
    lat2_rad = np.radians(lat2)
    lon2_rad = np.radians(lon2)
    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad
    a = np.sin(dlat / 2)**2 + np.cos(lat1_rad) * np.cos(lat2_rad) *
np.sin(dlon / 2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    distance = R * c
    return distance
```

Add a "Distance" column to the far right of the combination dataframe and use haversine to determine the distance between two points using the coordinates obtained from the combination results. The distance between the start and finish nodes is calculated, and the result is shown in Table 5.

Table 5. Dataframe Distance between Nodes

Initial Node	*End Node*	*Distance*
A	B	0.721666
A	C	0.969056
A	D	0.488750
A	E	0.348262
A	F	0.821422

4. Implementation Stage

At this stage there are three activities, which are described as follows.

a. Graph creation

The process of creating a graph functions as a path identifier on a map based on a collection of points/nodes and edges/sides of the path, where each edge is connected to one or two nodes. This is because graphs are used to represent discrete objects and the relationships between these objects(Gede, 2018). Of course, you need a route and a map to use to analyze which route is the closest.

Referring to Figure 2, route creation is done using the Folium library as a form of route visualization on the map. The first thing to do is define the coordinates of each node in dictionary form and define the branch points and their travel distances which are known from the distance search process. The short code is shown in Table 6.

Table 6. The Code Defines the Coordinates and Defines the Graph

```
nodes_coordinates = {
 'A': (-6.919792, 107.606014),
 'B': (-6.915593, 107.601029),
 'C': (-6.927422, 107.610256),
 'D': (-6.924132, 107.606715),
 'E': (-6.916932, 107.604728),
 'F': (-6.927167, 107.606442),
}
```

```
Graph_nodes =  {
    'A': [('G',0.06865401351946396)],
    'B': [('X', 0.07995765678973407),('W', 0.09688326322152979)],
    'C': [('P', 0.14602782197057906)],
    'D': [('K', 0.18835720901514427)],
    'E': [('S',0.43637261732217836 ), ('Y',0.12742801993764627)],
    'F': [('O', 0.0681578548022687), ('Q',0.06342159328376686)],
}
```

Table 6 on the left shows that A is connected to G, then B is connected to X and W, and so on. Meanwhile, Table 6 at the bottom shows the definitions of the coordinates of each node and so on (G-Z). After that, routes can be created using looping and using previously defined coordinates. Table 7 shows the code created to create a graph on the map.

Table 7. The Code Creates a Route on the Map

```
import folium
m = folium.Map(location=[0,0], zoom_start=2)
for node, connections in Graph_nodes.items():
    node_coord = node_coordinates[node]
    folium.Marker(
        location=node_coord, node_coordinates
        popup=node,
        icon=folium.Icon(color='blue')
    ).add_to(m)

    for edge in connections:
        connected_node = edge[0]
        connected_node_coord = node_coordinates[connected_node]
        folium.PolyLine(
            locations=[node_coord, connected_node_coord],
            color='black'
        ).add_to(m)
```

b. Search for euclidean values

Following the completion of the node combination procedure, the distances of every feasible path are looked up. By comparing location distances and using the A* algorithm, users can find the closest place by utilizing Euclidean distance, which is the difference between two locations in space and the basis for pendulum rotation(Marcelina & Yulianti, 2020). Euclidean is related to the Pythagorean Theorem. The application of the Euclidean formula is as follows.

$$dist(x,y) = \sqrt{\sum_{i-1}^{n}(x_i - y_i)^2} \quad (3)$$

Where:

xi (latitude) = latitude lines leading from the equator (0) to the south pole, or the equator to the north pole (angle 0-90 and 0 -90).

yi (longitude) = longitude is a horizontal line such as from the equator. The angle 0 (Greenwich) towards Hawaii is 0-180, while the opposite is from 0 to -180.

Based on Table 8, the code used to calculate the Euclidean value is using a heuristic function. The goal of each route is from node A back to node A. So node A can be defined as a goal node.

Table 8. The Code Looks for Euclidean Values

```
goal_node = 'A'
def heuristic(n):
    node_lat, node_lon = df.loc[df['Nodes'] == n, ['Latitude_Node',
'Longitude_Node']].values[0]
    goal_lat, goal_lon = df.loc[df['Nodes'] == goal_node,
['Latitude_Node', 'Longitude_Node']].values[0]
    return np.sqrt((node_lat - goal_lat) ** 2 + (node_lon -
goal_lon) ** 2)
```

Then calculations are carried out using the Python programming language for Euclidean values. Then a Euclidean value is generated for each node. The dataframe of the Euclidean values found is in Table 9.

Table 9. Euclidean Value Dataframe

Nodes	*Distance*
A	0.000000
B	0.006518
C	0.008730
D	0.004396
E	0.003136

c. Implementation of the A* algorithm for route optimization

A graph search is performed in order to determine a path from the starting node to the destination node. Tracing is usually done by following the edges of connecting links between nodes as seen in Figure 3. The steps for using A* can be carried out with selected sample nodes, namely P, Q, R, F, N, M, and O.

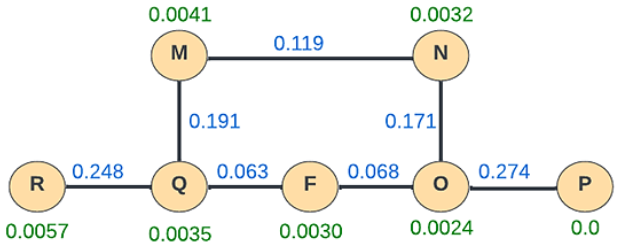

Figure 3. Graph Sample

The steps for implementing A* are as follows.

1. Initialization: node R is the origin and node P is the destination. The blue number is the distance or g(n) while the green number is the heuristic value or h(n).
2. Repeat the steps below:
   a. From the open set, find the node (n) with the lowest f(n) value. Specifically, the only path that may be taken is from R to Q.
   b. Move the point from the open set to the closed set or f(n) R-Q has been calculated and Q becomes the last node calculated.
   c. Check the neighbors of the selected point, namely between Q to R, Q to M or Q to F: (i) If it has never been checked, add it to the open set and calculate the cost so far (g(n)) and the estimated cost to the endpoint (h(n)). In this case, Q to M and Q to F have not been checked, so they will be checked. (ii) If it is already in open set, check if the new line is cheaper. If yes, update the path information. Q to M and Q to F are checked and Q to F has the cheapest path. (iii) If it is already in closed set, ignore it or check whether the new path is better. If so, move it back to open set and update the path information. In this case it is Q to R which has previously been calculated and used.
   d. There are no pathways possible if the open set is empty. Recreate the path from the end point to the beginning using the best path noted during the search if an end point is located.
   e. Repeat these steps until you find the end point.
3. Stop the loop if:
   a. If there are no more points in the "open set" list that need to be checked (meaning there are no possible paths).
   b. If the endpoint has been successfully found (then the shortest path has been found).

At this stage, if the graph and Euclidean values have been created, the next step is to apply the A* algorithm with the Python programming language.

Table 10. A* Formula Implementation Code in Python

```
while len (open_set) > 0:
  n = None
  for v in open_set:
      if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
           n = v
```

The code considers the actual costs that have been found (g(n)) and the expected remaining costs towards the objective (heuristic(n)) when determining the next node to be searched. It bases this comparison on Table 10 and the value f(n) of each node that is still being considered.

Determining a heuristic function that contains the Euclidean value for every node comes after the A* function has been run Table 11 shows a short code for defining the heuristic function.

Table 11. The Code Defines the Euclidean Value of Each Node

```
def heuristic(n):
        H_dist = {
        'A': 0.0,
'B': 0.006517808373989492,
'C': 0.008729917754483093,
'D': 0.004396248514357359,
'E': 0.003135824612445607,
```

To implement Dijkstra with the same sample, the following steps are taken.

1. Initialization: node R is the origin and node P is the destination. Set the initial distance to all nodes as infinity, except the distance to the origin node R which is set to zero.
2. Algorithm steps:
   a. From R, choose the unexplored node that is the closest in distance.
   b. Move the node from the unprocessed set to the visited set.
   c. Examine the node you have selected's neighbors: Determine the updated distance for each neighbor by using the chosen node to connect the neighbor node to the origin. Update the distance and save the neighbor node as the node to be visited if the new distance is less than the predetermined distance.
   d. Repeat steps a-c until all nodes have been visited or the shortest distance to the destination has been found.

The Dijkstra algorithm is implemented by the code in Table 14.

Table 12. Implementation of Dijkstra's Algorithm

```
def dijkstra(graph, start, goal):
    open_list = []
    heapq.heappush(open_list, (0, start))
    came_from = {}
    g_score = {node: float('inf') for node in graph}
    g_score[start] = 0
```

5. Evaluation Stage

At this stage, there are three activities, which are described as follows.

a. Comparison of algorithms

The A* method and the Dijkstra algorithm are compared concerning the compute time, CPU use, and the final route distance.

b. Creating a GUI (Graphical User Interface)

GUI creation is carried out with the aim of interactive use of computer software. Therefore, Python can be used to create GUIs that can be run in code editors. The Tkinter library is a standard GUI widget for creating Python interfaces. Tkinter is a graphics library that can make it easier to create graphics-based programs(Fajri et al., 2020). The code for importing the Tkinter library is in Table 13.

Table 13. The code imports the Tkinter library

```
import tkinter as tk
```

c. Displays the route on the map

To see an interactive visualization of the route taken, the Folium library is used to create a route line. The route line will be created from the departure route and the return route line to A.

RESULTS AND DISCUSSION

In order to determine the average value of each algorithm and increase the validity of the generated data, the route search algorithm can be evaluated by comparing the outcomes of ten algorithm routes that have the same beginning and ending places. The comparison results are in Table 14.

Table 14. Comparison of Route Finding Algorithms

Routes	*Time (ms)*		*CPU (%)*	
	*A**	*Dijkstra*	*A**	*Dijkstra*
1	0.000000	0.011999	4.40	4.00
2	0.001000	0.029999	4.70	5.80
3	0.000000	0.031000	3.50	9.20
4	0.000000	0.015000	5.10	3.30
5	0.001004	0.064006	3.50	7.30
6	0.001000	0.041246	4.80	5.40
7	0.000000	0.018998	8.90	3.10
8	0.001018	0.049004	9.90	21.00
9	0.000000	0.052006	6.60	10.20
10	0.000000	0.017002	4.20	3.60
Average	0.0004022	0.033026	5.56	7.29

The identical object from the starting node, the destination node, and the path and distance produced by both algorithms are used to compare A* and Dijkstra. Table 14 shows that A* can determine the nearest route search process time faster with an average time of 0.0004022 ms compared to Dijkstra which is 0.033026 ms. This is also supported by the results of the smallest average CPU usage, which is 5.56% compared to Dijkstra which is 7.29%.

This research implements a Graphical User Interface (GUI) which allows interactive visualization of routes on a map in a web browser. Through the application of mapping technology, this GUI allows users to enter starting points, stopping points and ending points, displaying the route visually. The evaluation carried out is to compare the distance between the available routes and the routes obtained from A*. The results and discussion of this research support the usefulness of this application in providing customizable route information and facilitating travel decision making and A* evaluation.

1. Graphical User Interface (GUI)

The GUI that has been designed is shown in Figure 4.

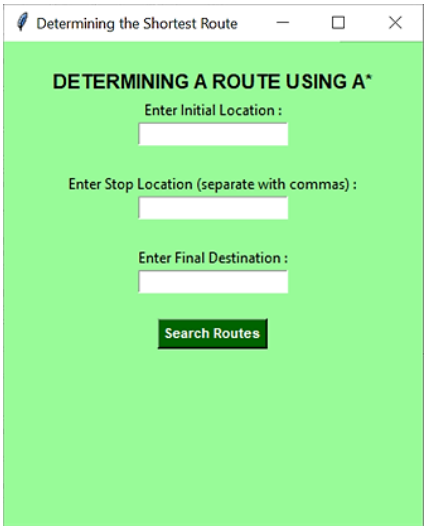

Figure 4. GUI to Determine the Closest Route to A*

In Figure 4, there is a filling form. The initial form is the initial node input. Then the second form can be used if there is more than one visit. The user can input other nodes. If there is more than one node, separate them with commas without spaces. Then for the last form is the final destination of the location. Then the "Search Route" button can be done after the form has been completely inputted. Filling out the form is as shown in Figure 5. Namely the starting location is A, then visiting S and L as stops, and the final destination is K.

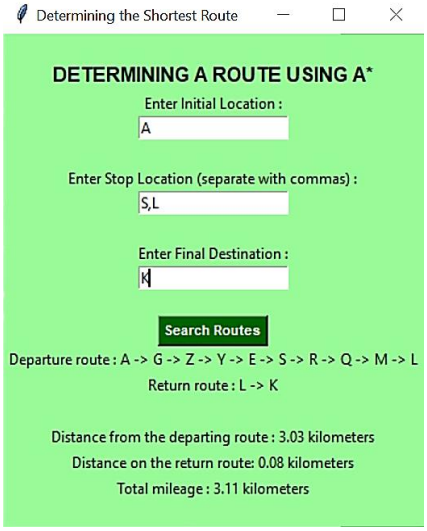

Figure 5. GUI after inputting location on form

Figure 5 shows that the shortest departure route obtained is A-G-Z-Y-E-S-R-Q-M-L as the departure route with a distance of 3.03 km, then continuing to L-K as the route to the final destination with a distance of 0.08 km. The total distance is the sum of the departure distance and the return distance, namely 3.11 km.

After filling in the form on the GUI and pressing the "Search Route" button, apart from the results listed on the GUI, another result is the visualization of the routes created by redirecting to the web browser.

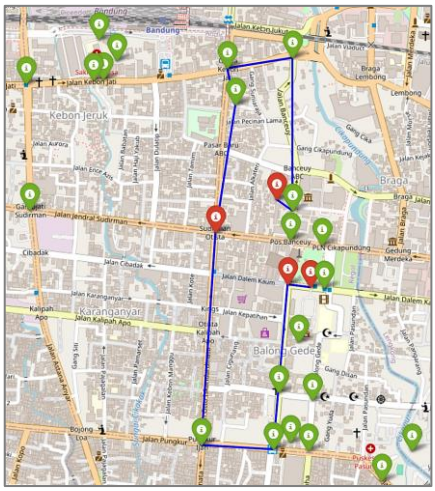

Figure 6. Routes created by A*

Figure 6 is the result of a route created using A*. Red markers indicate visited places namely A, S, L, and K, while green markers are other nodes. The blue line shows the path obtained to reach all nodes. The route is the result of visualization of the route findings according to the results listed on the GUI. Overall, the route becomes one road where the departure route is combined with the return route, namely A-G-Z-Y-E-S-R-Q-M-L-K.

2. Distance Evaluation

To evaluate whether the route determined by A* is correct with the shortest distance, a comparison is needed between the total distance of the route chosen by A* and other distance routes. There are several samples used for evaluation, namely the starting point at A, stopping points at S and L, and the final point at K. Some of the possible routes to fulfill the sample are as in Table 15.

Table 15. Possible routes and distances

No	*Sample*	*Rute*		*Distance (km)*
1	Sample 1 (A*)	Departure route	A-G-Z-Y-E-S-R-Q-M-L	3,03
		Return route	L-K	0,08
		Total distance		**3,11**
2	Sample 2	Departure route	A-G-Z-Y-E-S-R-Q-F-O-N-M-L	3,26
		Return route	L-K	0,08
		Total distance		**3,34**
3	Sample 3	Departure route	A-G-Z-Y-X-B-W-V-U-T-S-R-Q-M-L	4,46
		Return route	L-K	0,08
		Total distance		**4,54**
4	Sample 4	Departure route	A-G-Z-Y-X-B-W-V-U-T-S-R-Q-F-O-N-M-L-K	4,69
		Return route	L-K	0,08
		Total distance		**4,77**

From Table 15, based on the route from point A to S then to L, and finally to K, the route chosen by the A* algorithm has the shortest total distance, namely a route of 3.11 kilometers. Meanwhile, other routes have longer distances. Namely, Sample Route 2 has a distance of 3.34 km. Sample Route 3 has a distance of 4.54 km. Sample Route 4 has a distance of 4.77 km.

The study's findings offer a clearer knowledge of how the A* algorithm is used in the logistics sector, particularly in one of the cities of Bandung's logistics companies, to determine the quickest path for pickup and delivery services. The comparison between the route chosen by the A* algorithm and other routes demonstrates the effectiveness of the research findings, which point to the A* algorithm's ability to locate the shortest path. By using the A* algorithm, the resulting routes tend to be more

efficient in terms of distance traveled, which is an important factor in optimizing pickup and delivery services.

Aside from that, the planned GUI implementation was carried out effectively and should facilitate couriers in determining the shortest path while providing unambiguous information about the distance covered. The successful implementation of GUI shows the potential of technology to increase the operational efficiency of pickup and delivery services in the logistics industry.

CONCLUSIONS AND RECOMMENDATIONS

The most efficient route—in this example, the shortest one—for pickup and delivery services from a logistics company in Bandung City is determined by this study using the A* (A-star) algorithm. Compared to Dijkstra, the A* algorithm shows less CPU usage, which is 5.56% compared to Dijkstra, which is 7.29%. This is also supported by a faster computing time, which is 0.0004022 ms, compared to Dijkstra, which is 0.033026 ms. From the results obtained, if the user comes from position A, namely KCU Bandung, then stops at S and L, then the final point is at K, then the departure route generated using Algorithm A* is A–G–Z–Y–E–S–R–Q–M –L as the departure route and L–K route as the return route. The total distance found was 3.11 kilometers. Compared to other alternative routes, the route selected using the A* algorithm has a shorter route. Where Sample route 2 has a distance of 3.34 km. Sample Route 3 has a distance of 4.54 km. Sample Route 4 has a distance of 4.77 km. A comparison between the route chosen by the A* algorithm and other possible routes demonstrates the effectiveness of the algorithm in determining the shortest path.

However, this study also has several limitations. First, this research only considers the distance factor in determining the shortest route, while other factors such as travel time or traffic conditions are not considered. It is advised to conduct additional studies in the future to determine how well the A* algorithm performs while determining the shortest path while taking into account changes in traffic patterns or other barriers. Furthermore, it is possible to investigate the inclusion of variables like trip duration or fuel prices in order to present a more comprehensive image of the ideal route selection. In doing so, studies can offer a more thorough understanding of the benefits and constraints of the A* algorithm in actual situations with intricate traffic and mobility issues.

REFERENCES

Aswandi, & Marlina, L. (2023). Implementation of the A Star Heuristic Search Algorithm in Determining the Shortest Path. *International Journal of Computer Sciences and Mathematics Engineering*, *2*(1). https://doi.org/10.61306/ijecom.v2i1.20

Behún, M., Knežo, D., & Cehlár, M. (2022). Recent Application of Dijkstra's Algorithm in the Process of Production Planning. *Applies Sciences*, *12*. https://doi.org/10.3390/app12147088

Bismi, W., Gata, W., Anton, & Asra, T. (2021). Penerapan Algoritma Hybrid Dalam Menentukan Rute Terpendek Antara Cabang Kampus. *Ultima Computing*, *13*. https://doi.org/10.31937/sk.v13i1.1856

Boysen. Nils, Fedtke, S., & Schwerdfeger, S. (2020). Last-mile delivery concepts: A survey from an operational research perspective. *OR Spectrum*, *43*. https://doi.org/10.1007/s00291-020-00607-8

Candra, A., Budiman, M. A., & Pohan, R. I. (2021). Application of A-Star Algorithm on Pathfinding Game. *Journal of Physics: Conference Series*. https://10.0.4.64/1742-6596/1898/1/012047

Cano, J. A., Gómez-Montoya, R. A., Salazar, F., & Cortés, P. (2021). Disruptive and Conventional Technologies for the Support of Logistics Processes: A Literature Review. *International Journal of Technology (IJTech)*, *12*(3). https://doi.org/10.14716/ijtech.v12i3.4280

Dablanc, L., Diziain, D., & Levifve, H. (2011). Urban freight consultations in the Paris region. *European Transport Research Review*, *3*(1). https://etrr.springeropen.com/articles/10.1007/s12544-011-0049-2

Durdu, A., & Kaya, M. F. (2023). The Effects of Route Optimization Software to the Customer Satisfaction. *Sakarya University Journal of Science*, *27*(4). http://dx.doi.org/10.16984/saufenbilder.1259595

Fajri, Effendi, T. R., & Fadillah, N. (2020). Sistem Absensi Berbasis Pengenalan Wajah Secara Real Time menggunakan Metode Fisherface. *Jurnal Nasional Informatika Dan Teknologi Jaringan*, *4*(2). https://doi.org/10.30743/infotekjar.v4i2.2377.

Feng, B., & Ye, Q. (2021). Operations management of smart logistics: A literature review and future research. *Frontiers of Engineering Management*, *8*. https://doi.org/10.1007/s42524-021-0156-2.

Firwanda, A. Y., Prianto, C., & Rahayu, W. I. (2021). Penentuan Rute Terpendek Lokasi Badan Pusat Statistik Kota Bandung Dengan Algoritma Dijkstra. *JUTEKIN*, *9*. http://dx.doi.org/10.51530/jutekin.v9i1.509

Foead, D., Ghifari, A., Kusuma, M. B., Hanafiah, N., & Gunawan, E. (2021). A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science*, *179*, 507–514. https://doi.org/10.1016/j.procs.2021.01.034

Gede Wahyu Antara Dalem, I. B. (2018). Penerapan Algoritma A* (Star) Menggunakan Graph untuk Menghitung Jarak Terpendek. *Jurnal Resistor*, *1*(1), 41–47. https://dx.doi.org/10.31598/jurnalresistor.v1i1.253.

Jiang, L., & Mahmassani, H. S. (2014). City Logistics: Freight Distribution Management with Time-Dependent Travel Times and Disruptive Events. *Transportation Research Record: Journal of the Transportation Research Board*, *2410*(1). https://doi.org/10.3141/2410-10.

Liu, L., Wang, B., & Xu, H. (2022). Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method. *MDPI Electronics*, *11*(22). https://doi.org/10.3390/electronics11223660.

Marcelina, D., & Yulianti, E. (2020). Aplikasi Pencarian Rute Terpendek Lokasi Kuliner Khas Palembang Menggunakan Algoritma Euclidean Distance dan A*(Star). *Jurnal Sistem Informasi Dan Komputer*, *9*(2). https://doi.org/10.32736/sisfokom.v9i2.827.

Ngo, Q.-H. (2023). The effectiveness of market orientation in the logistic industry: A focus on SMEs in an emerging country. *Heliyon*, *9*(7). https://doi.org/10.1016/j.heliyon.2023.e17666

Palacin, J., Rubies, E., Bitria, R., & Clotet, E. (2023). Path Planning of a Mobile Delivery Robot Operating in a Multi-Story Building Based on a Predefined Navigation Tree. *Sensors*, *23*(21). https://doi.org/10.3390/s23218795.

Pane, S. F., Awangga, R. M., Rahcmadani, E. V., & Permana, S. (2019). Implementasi Algoritma Genetika Untuk Optimalisasi Pelayanan Kependudukan. *Jurnal Teknik Insentif*, *13*(2), 36–43. https://doi.org/https://doi.org/10.36787/jti.v13i2.130

Patel, M., & Patel, N. (2019). Exploring Research Methodology: Review Article. *International Journal of Research and Review*, *6*(3). https://www.ijrrjournal.com/IJRR_Vol.6_Issue.3_March2019/IJRR0011.pdf

Pranoto, F. S. D. (2020). Penggunaan Algoritma A-Star untuk Menentukan Rute Tercepat di dalam Kampus Ganesha ITB. *Makalah IF2120 Matematika Diskrit*.

Sheng, L., Bao, L., & Wu, P. F. (2018). Application of heuristic approaches in the robot path planning and optimization: a review. *Electron. Opt. Control*, *25*, 58–64.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, *104*, 333–339. https://doi.org/10.1016/j.jbusres.2019.07.039

Taherdoost, H. (2021). Data Collection Methods and Tools for Research; A Step-by-Step Guide to Choose Data Collection Technique for Academic and Business Research Projects. *International Journal of Academic Research in Management*, *10*(1), 10–38. https://elvedit.com/journals/IJARM/wp-content/uploads

Trivaika, E., & Senubekti, M. A. (2022). Perancangan Aplikasi Pengelola Keuangan Pribadi Berbasis Android. *Nuansa Informatika Technology and Information Journal*, *16*(1). https://doi.org/10.25134/nuansa.v16i1.4670

Wayahdi, M. R., Ginting, S. H. N., & Syahputra, D. (2021). Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path. *International Journal of Advances in Data and Information Systems*, *2*(1). http://dx.doi.org/10.25008/ijadis.v2i1.1206

Winkelhaus, S., & H. Grosse, E. (2019). Systematic Review Towards a New Logistics System. *International Journal of Production Research*, *58*(1). https://doi.org/10.1080/00207543.2019.1612964

Wulandari, I. A., & Sukmasetya, P. (2022). Implementasi Algoritma Dijkstra untuk Menentukan Rute Terpendek Menuju Pelayanan Kesehatan. *JISI*, *1*. http://dx.doi.org/10.24127/jisi.v1i1.1953

Zhu, J., & Zhu, Z. (2023). The space-time evolution and driving mechanism of coordinated development of modern logistics industry and tourism industry. *Journal of Cleaner Production*. https://doi.org/10.1016/j.jclepro.2023.138620