JOURNAL LA MULTIAPP

VOL. 06, ISSUE 02 (445-456), 2025
DOI: 10.37899/journallamultiapp.v6i2.2043

Learning to Optimize Meter Reading Routes in Billing Management Problems in the Balata Regional Unit

Gatot Mochamad Muchtar[1], Rinovia Mery Garnierita Simanjuntak[1]

[1]*Institut Teknologi Bandung*

**Corresponding Author: Gatot Mochamad Muchtar*
Email: 20923019@mahasiswa.itb.ac.id

Article Info

Article history:
Received 27 February 2025
Received in revised form 3 May 2025
Accepted 12 May 2025

Keywords:
Meter Reading Route
Traveling Salesman Problem
Optimization

Abstract

Meter reading route optimization is a significant challenge in the billing management of the State Electricity Company (PLN). This study evaluates the potential of the Traveling Salesman Problem (TSP) in overcoming this problem. By applying TSP to the Balata Region case study, this study aims to find a more efficient and effective route solution. The results of the study are expected to reduce operational costs, increase officer productivity, and increase customer satisfaction. It is hoped that TSP can be an attractive alternative for electricity companies in optimizing the meter reading process.

Introduction

Route optimization is one of the main challenges in various industrial sectors, including the energy sector. In the context of electricity service providers, optimizing meter reading routes is crucial to increasing operational efficiency, reducing costs and improving the quality of customer service. Traditionally, meter reading route planning is often done manually or using conventional algorithms such as the Nearest Neighbor algorithm or the Savings algorithm.

However, with the increasingly complex electricity distribution network and the increasing number of customers, this conventional approach is starting to show its limitations in producing optimal and flexible solutions. One of the classic problems in route optimization is the Traveling Salesman Problem (TSP) (Pop et al., 2024; Aranski, 2022; Mzili et al., 2022; Zhang & Yang, 2022). TSP is a combinatorial optimization problem that aims to find the shortest route that allows a traveling salesman to visit each city in the list once and return to the home city (Nemani et al., 2021; Hlaing & Khine, 2011; Rondano, 2025). According to Kovács & Jlidi (2024) the application of neural networks in the context of the Vehicle Routing Problem shows significant potential in improving the efficiency of route planning, which is also relevant in the context of TSP.

An electricity distribution network can be modeled as a graph, where each customer is a node and the relationships between customers (E.g., geographic distance) are represented as edges (Beinert et al., 2023; Vassallo et al., 2024; Das & Soylu, 2023; Peng et al., 2024). By utilizing this graphical structure, TSP can be used to find more optimal route solutions compared to conventional algorithms. Lischka et al. (2024) also emphasize the importance of efficient architectures for graph-based problems, such as TSP, which can improve algorithm performance in solving route optimization problems. Apart from that, TSP also has advantages in handling dynamic and incomplete data. In the context of meter reading, the condition of the electricity distribution network can change dynamically due to disruptions, changes in demand, or the addition of new customers (Zafeiropoulou et al., 2022; Gallegos et

ISSN: 2716-3865 (Print), 2721-1290 (Online)

al., 2024; Chen et al., 2023). The algorithm that solves TSP is able to adapt to these changes and produce a route solution that remains optimal. Previous research has demonstrated the potential of TSP in areas such as logistics, travel planning, and freight forwarding. Brody et al. (2021) also showed that attention in Graph Attention Networks can improve accuracy in solving graph-based problems, including TSP.

However, the application of TSP in meter reading route optimization is still relatively limited. Therefore, this research aims to fill this gap by comparing the performance of TSP with conventional algorithms in the context of meter reading route optimization. As a case study, this research will focus on electricity customers in the Balata Region as a research object based on unique customer characteristics and routes with not many customers, but the working area is quite large and dominated by hills, as well as the availability of data that can be retrieved at any time.

Thus, it is hoped that the results of this research can make a significant contribution to increasing the operational efficiency of electricity companies and provide inspiration for further research in the same field. This research will also consider various machine learning algorithms that have been proven effective in data processing, such as Random Forest (Breiman, 2001), Support Vector Networks (Cortes, 1995), dan Ensemble Machine Learning (Natras et al., 2022). In addition, the deep learning approach proposed by Lecun et al. (2015) and previous research on predictions using machine learning algorithms (Punnoose & Xlri - Xavier, 2016; Şahinbaş, 2022) will be an important reference in the development of this model.

Methods

The methodology in this code involves Geographical Information Systems (GIS) which is used for geospatial data processing and Graph Analysis. Road networks are modeled as graphs, road network analysis to determine optimal routes in operational applications. The initial step is collecting customer data which is stored in an Excel file, which contains information such as customer ID, latitude and longitude. This data is validated to ensure the relevant columns meet program requirements. Next, the geospatial space is modeled by calculating the maximum and minimum coordinates from customer data to determine the operating area on the map. Using OSMnx, the road network in this area is downloaded from OpenStreetMap in drive mode, so that it only includes roads that can be passed by vehicles.

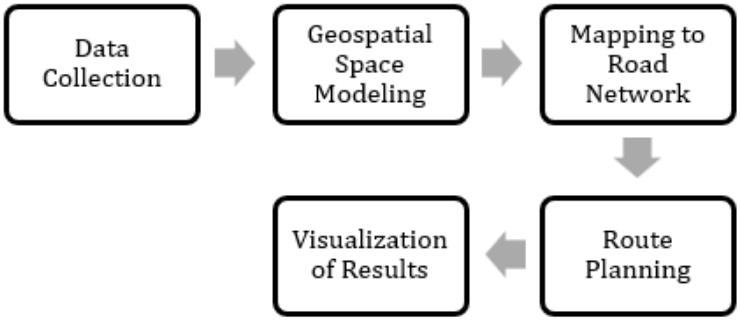


Figure I. Processing and analysis of road networks based on geospatial data

After that, each customer is mapped to the nearest node on the road network using the ox.distance.nearest_nodes() function. Route planning is carried out by selecting the initial node from customer data and determining the order of visits using the greedy nearest neighbor approach, which prioritizes the shortest distance to the next node based on the shortest path algorithm. In this process, the nearest neighbor algorithm is used to select the next node by considering the closest distance along the road network. The final result is visualized using Folium in the form of an interactive map, where customer points are marked with markers

ISSN: 2716-3865 (Print), 2721-1290 (Online)

and paths between customers are drawn using red lines. This map is saved as an HTML file for further analysis and is also displayed in the program and can be downloaded as an excel file.

The first stage is collecting customer data which is stored in an Excel file. This file contains information such as customer ID (IDPEL), latitude coordinates (LATITUDE), and longitude (LONGITUDE). The uploaded data is then validated to ensure the relevant columns meet program requirements. This validation is important so that the subsequent processing process can run without errors. To model the operating space, the maximum and minimum coordinates of the customer data are calculated to determine the relevant map areas. This area is used as a bounding box to retrieve the road network from OpenStreetMap using OSMnx. By selecting drive mode, the road network taken only includes roads that can be passed by vehicles, making it more relevant for operational needs. At this stage, each customer point is mapped to the nearest node in the road network. This process is carried out with the ox.distance.nearest_nodes() function which matches the customer's latitude and longitude coordinates with the nearest node in the road network graph. Thus, customer locations are linked directly to road routes in the network model.

Route determination is done by selecting the initial node from customer data, then using the greedy nearest neighbor approach. This algorithm works by selecting the next node based on the shortest distance from the node being visited, which is calculated using the shortest path algorithm. With this algorithm, optimal routes are generated based on the closest connections between nodes, thereby minimizing travel distance. The processing results are visualized using Folium in the form of an interactive map. Each customer point is displayed as a marker with customer ID information, while the routes between customers are drawn as red lines. This visualization helps verify and analyze the planned route results. The interactive map is then saved as an HTML file for further use

Results and Discussion

The results of research carried out to optimize customer visit routes using the clustering method and the Traveling Salesman Problem (TSP) algorithm. This research aims to group customers based on their geographic location and determine an efficient visit sequence for each group. By using geographic data, this research is expected to provide better solutions in planning visit routes, which in turn can increase operational efficiency and reduce travel costs. The data used in this research was taken from an Excel file uploaded by the user. The data consists of the following columns:

LATITUDE : Latitude of the customer's location.
LONGITUDE : Longitude of the customer location.
IDPEL : Customer identification.

Once the file is uploaded, the data is read using the Pandas library and prepared for further analysis. This process involves data quality checks, including checking for missing values and data format validation. Programming was carried out on Google Colabs using the Python programming language and using several libraries with the following results: The following is an explanation of the Python code in Figure 2 which is used to solve the Traveling Salesman Problem (TSP) problem using the nearest neighbor algorithm and clustering using KMeans. The code also manages geographic data from Excel files and generates optimal routes for multiple people based on customer location.

Import Library

Pandas and numpy: Used for data manipulation and numeric operations. KMeans from sklearn.cluster: Used to perform clustering on data. Files from google.colab: Used to upload and download files in Google Colab.

ISSN: 2716-3865 (Print), 2721-1290 (Online)

math for trigonometric functions needed in distance calculations.

Function haversine_distance

Calculate the distance between two points on the earth's surface based on latitude and longitude coordinates using the Haversine formula. Returns the distance in kilometers.

Fungsi tsp_nearest_neighbor

Implementing the nearest neighbor algorithm to solve TSP. Start from a starting point and iteratively search for the nearest unvisited point until all points have been visited. Returns the index sequence of visited points.

Upload Excel File

Use files.upload() to upload an Excel file containing customer data.

Read Data from Excel File

Use pd.read_excel() to read data from the uploaded file. Ensure that the columns used for latitude, longitude, and customer ID match the column names in the file.

Input number of people and reading days

Takes input from the user for the number of people who will visit and the number of reading days.

Customer Clustering

Using KMeans to group customers based on latitude and longitude coordinates. Added a 'cluster' column to the DataFrame to indicate the group of each customer.

Running TSP for Each Cluster

For each cluster, the tsp_nearest_neighbor function is called to obtain the visit sequence. Create a result DataFrame for each cluster and populate the 'Order of Visits' column based on the order visited.

Calculating Distance and Working Time

Calculates distance and working time for each visit based on the resulting sequence. Uses average speed (e.g. 40 km/h) to calculate travel time.

Calculating the Number of Reading Days

Calculates the number of visits per day based on the total visits and the specified number of reading days. Fill in the 'Number of Reading Days' column with information on reading days for each visit.

Combining Results

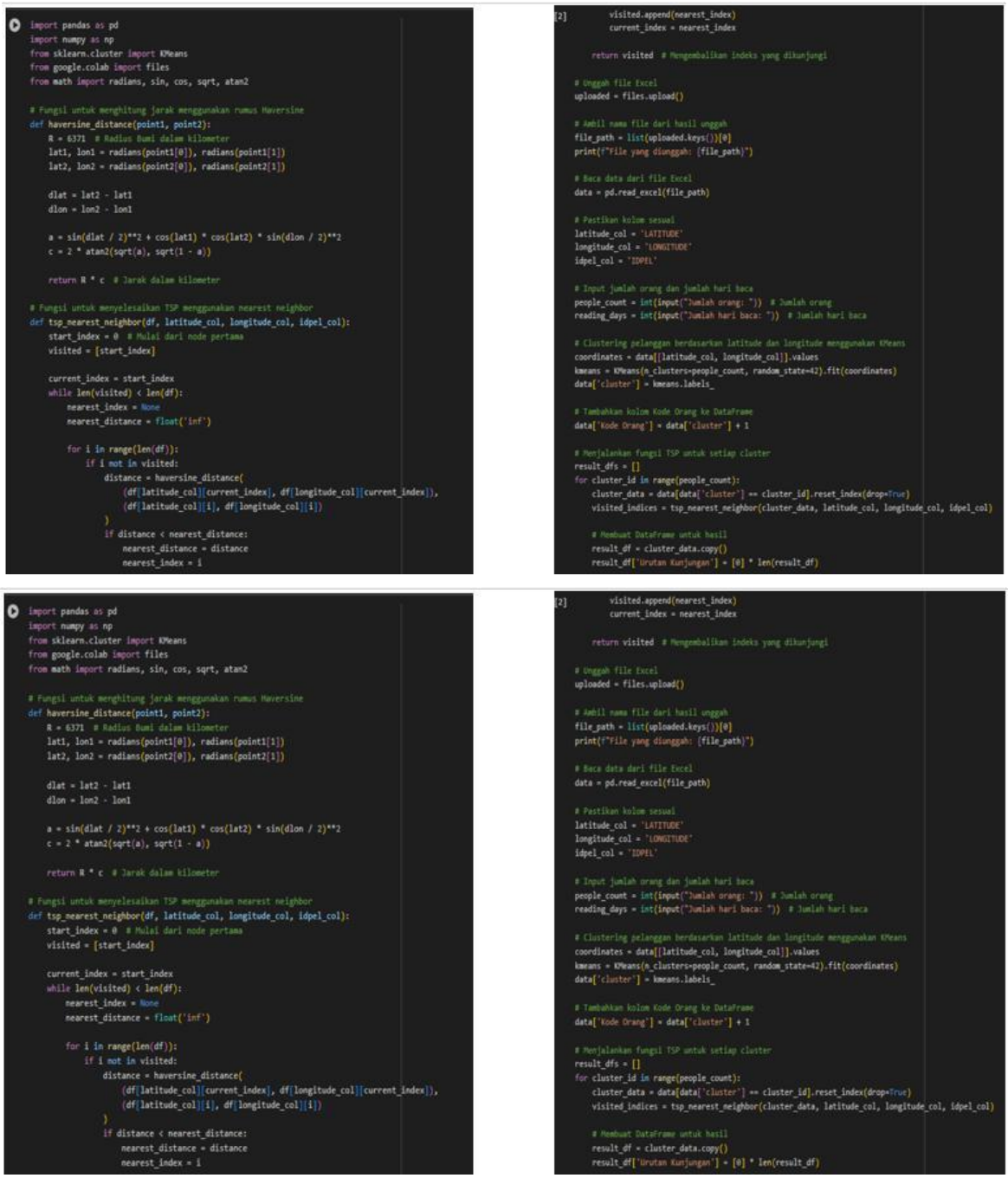
Combines all the resulting DataFrames from each cluster into one final DataFrame.

Saving Results to Excel File

Save the final DataFrame to an Excel file with the name 'optimal_route_with_clusters_and_read_days.xlsx'. Use files.download() to download a saved Excel file.

The overall aim of this code is to optimize customer visit routes based on their geographic location, taking into account the number of people making visits and the number of days available for making visits. By using clustering and the TSP algorithm, this code produces efficient and organized routes for everyone involved.

ISSN: 2716-3865 (Print), 2721-1290 (Online)



```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from google.colab import files
from math import radians, sin, cos, sqrt, atan2

# Fungsi untuk menghitung jarak menggunakan rumus Haversine
def haversine_distance(point1, point2):
    R = 6371  # Radius Bumi dalam kilometer
    lat1, lon1 = radians(point1[0]), radians(point1[1])
    lat2, lon2 = radians(point2[0]), radians(point2[1])

    dlat = lat2 - lat1
    dlon = lon2 - lon1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    return R * c  # Jarak dalam kilometer

# Fungsi untuk menyelesaikan TSP menggunakan nearest neighbor
def tsp_nearest_neighbor(df, latitude_col, longitude_col, idpel_col):
    start_index = 0  # Mulai dari node pertama
    visited = [start_index]

    current_index = start_index
    while len(visited) < len(df):
        nearest_index = None
        nearest_distance = float('inf')

        for i in range(len(df)):
            if i not in visited:
                distance = haversine_distance(
                    (df[latitude_col][current_index], df[longitude_col][current_index]),
                    (df[latitude_col][i], df[longitude_col][i])
                )
                if distance < nearest_distance:
                    nearest_distance = distance
                    nearest_index = i
```

```
[2]         visited.append(nearest_index)
            current_index = nearest_index

        return visited  # Mengembalikan indeks yang dikunjungi

# Unggah file Excel
uploaded = files.upload()

# Ambil nama file dari hasil unggah
file_path = list(uploaded.keys())[0]
print(f"File yang diunggah: {file_path}")

# Baca data dari file Excel
data = pd.read_excel(file_path)

# Pastikan kolom sesuai
latitude_col = 'LATITUDE'
longitude_col = 'LONGITUDE'
idpel_col = 'IDPEL'

# Input jumlah orang dan jumlah hari baca
people_count = int(input("Jumlah orang: "))  # Jumlah orang
reading_days = int(input("Jumlah hari baca: "))  # Jumlah hari baca

# Clustering pelanggan berdasarkan latitude dan longitude menggunakan KMeans
coordinates = data[[latitude_col, longitude_col]].values
kmeans = KMeans(n_clusters=people_count, random_state=42).fit(coordinates)
data['cluster'] = kmeans.labels_

# Tambahkan kolom Kode Orang ke DataFrame
data['Kode Orang'] = data['cluster'] + 1

# Menjalankan fungsi TSP untuk setiap cluster
result_dfs = []
for cluster_id in range(people_count):
    cluster_data = data[data['cluster'] == cluster_id].reset_index(drop=True)
    visited_indices = tsp_nearest_neighbor(cluster_data, latitude_col, longitude_col, idpel_col)

    # Membuat DataFrame untuk hasil
    result_df = cluster_data.copy()
    result_df['Urutan Kunjungan'] = [0] * len(result_df)
```

Figure 2. Clustering and TSP coding

The following is an explanation of the Python code provided, which aims to visualize customer visit routes based on location data uploaded from an Excel file:

Uploading and Reading Excel Files

Use files.upload() to upload Excel files to Google Colab. Read the Excel file into a DataFrame using pd.read_excel().

Setting Up Columns

Defines the column names to be used, such as LATITUDE, LONGITUDE, IDPEL, Day Code, Visit Order, and Person Code. Ensure that the required columns are present in the DataFrame. If any column is missing, a warning message will be printed.

Defining a Bounding Box

ISSN: 2716-3865 (Print), 2721-1290 (Online)

Calculates the north, south, east, and west boundaries of location data by adding a 0.1 degree buffer for each direction. This is used to define the area to be downloaded from OpenStreetMap.

Downloading Road Network

Using osmnx to download the road network within the area specified by the bounding box. This road network will be used to calculate the shortest route between visiting points.

Adding Nearby Nodes

Adds the nearest_node column to the DataFrame, which contains the nearest node in the road network for each customer location.

Sorting Data

Sorts data by Person Code and Visit Order if these columns exist. This ensures that the visiting routes are sequenced correctly.

Creating Maps with Folium

Initialize a map using folium.Map with a center location based on the average latitude and longitude of the data. Define a different color for each Person Code to differentiate routes.

Drawing Routes and Adding Markers

For each Person Code, draw a route based on the order of visits using the shortest path calculated with networkx. Add route lines to the map with folium.PolyLine. Add markers for each customer location with detailed information such as IDPEL, Person Code, Day Code, and Order of Visits.

Saving and Downloading Maps

Save the map that has been created into an HTML file. Download the HTML file so it can be viewed in a browser. This code as a whole aims to visualize customer visit routes based on uploaded location data, using the road network from OpenStreetMap to calculate the shortest route.

Figure 3. Map Image Coding

ISSN: 2716-3865 (Print), 2721-1290 (Online)

Data Quality Check

Data quality checks are carried out to ensure that the data used is valid and reliable. Some of the steps taken include: 1) Checks for missing values in the LATITUDE, LONGITUDE, and IDPEL columns; 2) Ensure that all latitude and longitude values are within valid ranges (latitude: -90 to 90, longitude: -180 to 180). Analysis is carried out using the KMeans algorithm to group customers based on their geographic coordinates. The number of clusters is determined by the user by inputting the number of people who will visit. After clustering, each customer is assigned a cluster label and person code.

Customer Clustering

Grouping is done using KMeans, which divides the data into a number of clusters according to the specified number of people. The clustering results are shown in the following table:

Table 1. Clustering Results

IDPEL	LATITUDE	LONGITUDE	People Code
121130047950	2.833141111	99.02620111	1
121130043040	2.833631111	99.02646111	1
121130043163	2.833681111	99.02642111	1
121130043784	2.833861111	99.02634111	1
121130043187	2.833971111	99.02628111	1
121130043112	2.834051111	99.02622111	1
121130135613	2.834151111	99.02636111	1

Clustering results show that customers are grouped based on their geographic proximity. Each cluster represents a group of customers that can be visited by one person in one route, in the table.

TSP solution

After clustering, the TSP algorithm is applied to each cluster to determine the optimal visit sequence. The algorithm used is the nearest neighbor method, which is a heuristic approach to solving the TSP problem. The results of the sequence of visits for each cluster are presented in Figure 5 below:

Table 2. Results of Visit Sequence

IDPEL	Order of Visits	Distance (km)	Working Hours (hours)
121130047950	0	0	0
121130043040	1	0.061664061	0.001541602
121130043163	2	0.007116553	0.000177914
121130043784	3	0.021898445	0.000547248
121130043187	4	0.013928779	0.000348219
121130043112	5	0.011114599	0.000277865
121130135613	6	0.019115208	0.000477881
121130043148	7	0.024047496	0.000601187
121130043148	8	0.008094366	0.000202359
121130043077	9	0.009839436	0.000245986
121130047935	10	0.02468873	0.000617218
121130043065	11	0.01480418	0.00037062
121130043151	12	0.019163434	0.000479086
121130043136	13	0.01143752	0.000285938
121130043772	14	0.063342638	0.001583566

ISSN: 2716-3865 (Print), 2721-1290 (Online)

Calculation of Distance and Working Time

After determining the order of visits, distance and working time calculations are carried out. Distances are calculated using the Haversine formula, which takes into account the curvature of the earth. Working time is calculated based on the specified average speed (here used, 40 km/h). Figure 6 below shows the total distance and working time for each cluster:

Table 3. Distance and total time results

People Code	Total Distance (km)	Total Working Time (hours)
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152
1	125.5296061	3.138240152

From Table 3 Total distance and time, it can be seen that the total distance and time are based on the average division of working time per clustering, for officers one day works at a distance of 125.52 km with a working time of 3.14 hours, but this is using average travel data. , for the entire excel download results can be seen in Figure 4.

Table 4. Excel download results

IDPEL	Latitude	Longitude	Ppl Code	Ordr of Visits	Distance (km)	Working Hours (hours)	Total Distance (km)	Working Time (hours)	Number of Days Read
121130090106	2.678061	98.927181	1	1	0.0194	0.00097	36.89	1.8448	Reading Day 1
121130080217	2.678221	98.927251	1	2	0.0141	0.00079	36.89	1.8448	Reading Day 1
121130091069	2.680176	98.926909	1	14	0.0706	0.0035	36.89	1.8448	Reading Day 2
121130080160	2.680211	98.926709	1	15	0.0226	0.0011	36.89	1.8448	Reading Day 3
121130108164	2.692951	98.927231	1	157	0.5302	0.0270	36.89	1.8448	Reading Day 4
121130108078	2.702961	98.920751	1	158	1.5244	0.0663	36.89	1.8448	Reading Day 5
121130081984	2.721701	98.933911	1	216	0.0244	0.0012	36.89	1.8448	Reading Day 6
121130131717	2.731911	98.935811	1	217	2.4739	0.1237	36.89	1.8448	Reading Day 7
121130047950	2.833141	99.026201	2	1	0.0	0.0	125.39	6.2695	Reading Day 1
121130043040	2.833631	99.026411	2	2	0.0616	0.0031	125.39	6.2695	Reading Day 1
121130005966	2.895411	99.038011	2	405	0.0156	0.00078	125.39	6.2635	Reading Day 2
121130126438	2.849861	99.038195	2	406	0.0375	0.00188	125.39	6.2695	Reading Day 3
121130000203	2.859885	99.025908	2	61	0.0083	0.00042	125.39	6.2695	Reading Day 4
121130015340	2.855201	99.023161	2	958	0.0081	0.0004	125.39	6.2695	Reading Day 5
121130005760	2.865231	99.023151	2	959	0.0035	0.00018	125.39	6.2695	Reading Day 6
121130045252	2.878251	99.104711	2	1000	0.0151	0.00076	125.39	6.2695	Reading Day 7
121130135394	2.573729	98.978586	3	1	62.36	3.118	62.36	3.118	Reading Day 1
121130113085	2.574929	98.978418	3	2	0.1341	0.0067	62.36	3.118	Reading Day 1
121130104056	2.610891	98.956841	3	161	0.1131	0.0056	62.36	3.118	Reading Day 2
121130104378	2.612041	98.956731	3	162	0.1394	0.0070	62.36	3.118	Reading Day 3
121130115318	2.618591	98.974621	3	217	1.1263	0.0563	62.36	3.118	Reading Day 4

ISSN: 2716-3865 (Print), 2721-1290 (Online)

121130131211	2.619521	98.975471	3	218	0.1400	0.0070	62.36	3.118	Reading Day 5
121130104936	2.635038	99.061693	3	627	0.0843	0.0047	62.36	3.118	Reading Day 6
121130135706	2.635448	99.071284	3	628	0.0644	0.0032	62.36	3.118	Reading Day 7
121130133007	2.747813	98.970636	4	1	65.84	3.2921	65.84	3.2921	Reading Day 1
121130120292	2.747963	98.970911	4	2	0.0348	0.0017	65.84	3.2921	Reading Day 1
121130038327	2.785031	98.936101	4	167	0.0122169	0.0006109	65.841747	3.292087	Reading Day 2
121130044766	2.785261	98.936131	4	168	0.0257909	0.0012895	65.841747	3.292087	Reading Day 3
121130062407	2.804161	98.936861	4	410	0.1637857	0.0811892	65.841747	3.292087	Reading Day 4
121130131676	2.810281	98.938011	4	411	1.1592338	0.0573617	65.841747	3.292087	Reading Day 5
121130021714	2.818711	98.960201	4	658	0.0153747	0.0007687	65.841747	3.292087	Reading Day 6
121130040411	2.818821	98.961311	4	659	0.0144973	0.0007246	65.841747	3.292087	Reading Day 7
121130064086	2.784661	99.020741	5	2	0.0120282	0.0006012	121.549063	6.077453	Reading Day 1
121130049608	2.784751	99.020801	5	2	0.0120282	0.0006012	121.549063	6.077453	Reading Day 1
121130035737	2.809321	99.030251	5	204	0.0159649	0.0007848	121.549063	6.077453	Happy Back 2
121130017199	2.809611	99.030311	5	205	0.0392278	0.0016470	121.549063	6.077453	Happy Back 3
121130068240	2.822191	99.072171	5	571	0.0113265	0.0005663	121.549063	6.077453	Hari Baca 4
121130068434	2.822211	99.072171	5	572	0.0070249	0.0003512	121.549063	6.077453	Hari Baca 5
121130132404	2.838231	99.052981	5	2145	0.0100568	0.0005028	121.549063	6.077453	Hari Baca 6
121130055087	2.850365	99.052985	5	2146	1.3493260	0.0674683	121.549063	6.077453	Hari Baca 7
121130111861	2.622811	98.938681	6	648	0.0136290	0.0006841	106.748703	5.337435	Reading Day 1
121130116877	2.622931	98.938911	6	2	0.0328256	0.0164128	106.748703	5.337435	Reading Day 1
121130112455	2.658191	98.937211	6	648	0.0136290	0.0006841	106.748703	5.337435	Reading Day 2
121130075255	2.658851	98.939351	6	649	0.0739066	0.0036953	106.748703	5.337435	Reading Day 3
121130081492	2.663601	98.938401	6	820	0.0233568	0.0011678	106.748703	5.337435	Reading Day 4
12113008254	2.663651	98.938441	6	821	0.0078584	0.0003929	106.748703	5.337435	Reading Day 5
121130109384	2.656523	98.935009	6	988	0.0999884	0.0049984	106.748703	5.337435	Reading Day 6
121130085337	2.667791	98.937401	6	989	0.2092729	0.0140636	106.748703	5.337435	Reading Day 7

Table 4, the downloaded Excel results present data related to customer visits, including several important columns that support operational analysis. The "IDPEL" column uniquely identifies the customer, while "LATITUDE" and "LONGITUDE" indicate their geographic location. “Person Code” and “Visiting Sequence” help in organizing and tracking the route of visits made by officers. Additionally, the table also includes information regarding “Distance (km)” and “Working Time (hours)” for each visit, which is useful in measuring operational efficiency. The columns "Total Distance (km)" and "Total Working Time (hours)" provide a cumulative overview of the activity, while "Number of Reading Days" records the frequency of visits in a certain period. This data is invaluable for route planning, time optimization, and effective resource management.

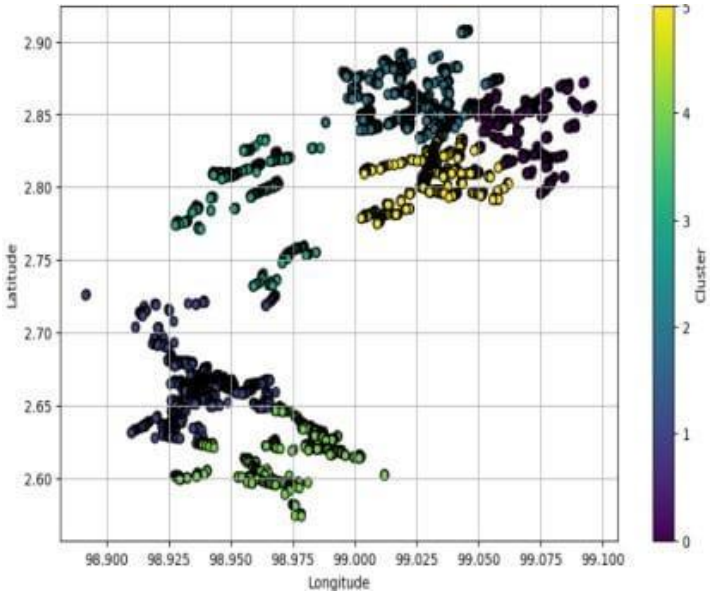


Figure 4. Customer Graph for Balata Region

ISSN: 2716-3865 (Print), 2721-1290 (Online)

The research results show that customer grouping using KMeans and TSP completion can optimize visit routes. The total distance and working time for each cluster are calculated and presented in Table 4. Clusters with a larger number of customers tend to have higher total distance, but working time does not always increase proportionally. The use of the TSP algorithm helps in reducing the total distance that must be traveled, which has implications for saving costs and time.

Visualization of results can be done using graphs or maps to show the route taken by each person. The map can show customer locations and the order of visits, providing a clear picture of the efficiency of the resulting route, the graph makes it easier to validate compliance with faster running times, the following graphic image can be seen in figure 5. Parapet ulp customer graph, for the map in use osmnx as ox and falium so that it can show the fastest route that can be taken with a display like in figure 6 Balata Regional Officer Route Map.

Figure 4 is a graph that depicts a visualization of Balata Region customers based on their geographic coordinates, with the horizontal axis showing longitude and the vertical axis showing latitude. Each point on the graph represents a customer location, and different colors indicate customer groupings (clusters) based on their geographic proximity.

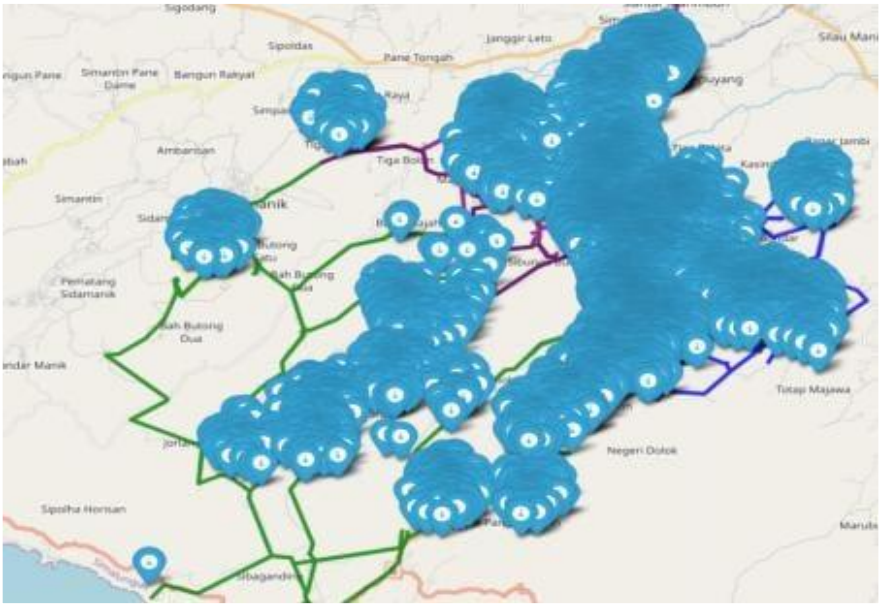


Figure 5. Route Map of Balata District Officers

Figure 5, Route Map of Balata Regional Officers shows the route taken by Balata Regional officers in carrying out their duties visiting customers. Each blue dot on the map represents the location a customer visited, while the line connecting the dots shows the path taken by the officer. Different line colors can indicate each officer's route. This visualization is very useful for analyzing route efficiency

Conclusion

The analysis results show that the method used is effective in grouping customers and determining the optimal order of visits. The use of the Haversine algorithm to calculate the distance between location points provides good accuracy in distance calculations. In addition, dividing the number of reading days based on the order of visits allows for more efficient scheduling. Integrating real-time data to improve the accuracy of distance and time calculations that will be carried out in Traveling Salesman Problem (TSP) programming. Develop web or mobile based applications that can help users plan visit routes automatically by adding TSP to get more optimal accuracy. Conduct further studies by considering other

ISSN: 2716-3865 (Print), 2721-1290 (Online)

factors such as time of visit and type of customer in creating the TSP model. With the addition of these details,. Can adjust tables and data according to the results obtained from the code being run, as well as adding relevant visualizations to enrich the presentation of research results.

Acknowledgment

The completion of this research was made possible thanks to the support and contributions of various individuals and institutions. We would like to express our deepest gratitude to: State Electricity Company (PLN) UID NORTH SUMATRA especially UP3 Pematangsiantar for providing data and resources which are very important for this research, as well as continuous support and cooperation throughout the research process. Bandung Institute of Technology (ITB), which has provided the academic environment and facilities that support this research activity.

References

Aranski, A. W. (2022). Optimization Of The Smallest Road Using The Traveling Salesman Problem (Tsp) Method. *IJISTECH (International Journal of Information System and Technology)*, *6*(1), 159-166. https://doi.org/10.30645/ijistech.v6i1.224

Beinert, D., Holzhüter, C., Thomas, J. M., & Vogt, S. (2023). Power flow forecasts at transmission grid nodes using graph neural networks. *Energy and AI*, *14*, 100262. http://dx.doi.org/10.1016/j.egyai.2023.100262

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. http://dx.doi.org/10.1023/A:1010950718922

Brody, S., Alon, U., & Yahav, E. (2021). How attentive are graph attention networks? *ArXiv Preprint ArXiv:2105.14491*. http://dx.doi.org/10.48550/arXiv.2105.14491

Chen, Z., Amani, A. M., Yu, X., & Jalili, M. (2023). Control and optimisation of power grids using smart meter data: A review. *Sensors*, *23*(4), 2118. https://doi.org/10.3390/s23042118

Cortes, C. (1995). Support-Vector Networks. *Machine Learning*.

Das, R., & Soylu, M. (2023). A key review on graph data science: The power of graphs in scientific studies. *Chemometrics and Intelligent Laboratory Systems*, *240*, 104896. https://doi.org/10.1016/j.chemolab.2023.104896

Gallegos, J., Arévalo, P., Montaleza, C., & Jurado, F. (2024). Sustainable electrification—advances and challenges in electrical-distribution networks: a review. *Sustainability*, *16*(2), 698. https://doi.org/10.3390/su16020698

Hlaing, Z. C. S. S., & Khine, M. A. (2011). Solving traveling salesman problem by using improved ant colony optimization algorithm. *International Journal of Information and Education Technology*, *1*(5), 404-409. http://dx.doi.org/10.7763/IJIET.2011.V1.67

Kovács, L., & Jlidi, A. (2024). Neural Networks for Vehicle Routing Problem. *ArXiv Preprint ArXiv:2409.11290*. http://dx.doi.org/10.32971/als.2024.014

Lischka, A., Wu, J., Chehreghani, M. H., & Kulcsár, B. (2024). A GREAT Architecture for Edge-Based Graph Problems Like TSP. *ArXiv Preprint ArXiv:2408.16717*. http://dx.doi.org/10.48550/arXiv.2408.16717

Mzili, T., Riffi, M. E., Mzili, I., & Dhiman, G. (2022). A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem. *Decision making: applications in management and engineering*, *5*(2), 287-299. http://dx.doi.org/10.31181/dmame0318062022m

ISSN: 2716-3865 (Print), 2721-1290 (Online)

Natras, R., Soja, B., & Schmidt, M. (2022). Ensemble machine learning of random forest, AdaBoost and XGBoost for vertical total electron content forecasting. *Remote Sensing*, *14*(15), 3547. https://doi.org/10.3390/rs14153547

Nemani, R., Cherukuri, N., Rao, G. R. K., Srinivas, P. V. V. S., Pujari, J. J., & Prasad, C. (2021, November). Algorithms and optimization techniques for solving tsp. In *2021 Fifth international conference on I-SMAC (IoT in social, mobile, analytics and Cloud)(I-SMAC)* (pp. 809-814). IEEE. http://dx.doi.org/10.1109/I-SMAC52330.2021.9640907

Peng, J., Kimmig, A., Wang, D., Niu, Z., Liu, X., Tao, X., & Ovtcharova, J. (2024). Energy consumption forecasting based on spatio-temporal behavioral analysis for demand-side management. *Applied Energy*, *374*, 124027. https://doi.org/10.1016/j.apenergy.2024.124027

Pop, P. C., Cosma, O., Sabo, C., & Sitar, C. P. (2024). A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research*, *314*(3), 819-835. https://doi.org/10.1016/j.ejor.2023.07.022

Punnoose, R., & Xlri -Xavier, C. (2016). Prediction of Employee Turnover in Organizations using Machine Learning Algorithms A case for Extreme Gradient Boosting. In *IJARAI) International Journal of Advanced Research in Artificial Intelligence* (Vol. 5, Issue 9). www.ijarai.thesai.org

Rondano, F. (2025). *Heuristic Algorithms for the Min-Max Traveling Salesman Problem* (Doctoral dissertation, Politecnico di Torino).

Şahinbaş, K. (2022). Employee promotion prediction by using machine learning algorithms for imbalanced dataset. *2022 2nd International Conference on Computing and Machine Intelligence (ICMI)*, 1–5. http://dx.doi.org/10.1109/ICMI55296.2022.9873744

Vassallo, M., Leerschool, A., Bahmanyar, A., Duchesne, L., Gerard, S., Wehenkel, T., & Ernst, D. (2024). An Optimization Algorithm for Customer Topological Paths Identification in Electrical Distribution Networks. *arXiv preprint arXiv:2409.09073*. https://doi.org/10.48550/arXiv.2409.09073

Zafeiropoulou, M., Mentis, I., Sijakovic, N., Terzic, A., Fotis, G., Maris, T. I., ... & Ekonomou, L. (2022). Forecasting transmission and distribution system flexibility needs for severe weather condition resilience and outage management. *Applied Sciences*, *12*(14), 7334. https://doi.org/10.3390/app12147334

Zhang, Z., & Yang, J. (2022). A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization. *Computers & Industrial Engineering*, *169*, 108157. http://dx.doi.org/10.1504/IJCI.2014.064853

ISSN: 2716-3865 (Print), 2721-1290 (Online)