ROUTING OPTIMIZATION FOR AERONAUTICAL NETWORKS

by

Kumar Malay 21BLC1015

Adnan Ghani 21BLC1097

Mrinal Naveen 21BLC1023

A project report submitted to

Dr Markkadan S

SCHOOL OF ELECTRONICS ENGINEERING(SENSE)

in partial fulfilment of the requirements for the course of

BCSE308P – Computer Networks

in

B. Tech. ELECTRONICS AND COMPUTER ENGINEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

JULY, 2023

BONAFIDE CERTIFICATE

Certified that this project report entitled

"**Routing optimization for aeronautical networks**"

is a bonafide work of

Kumar Malay – 21BLC1015

Adnan Ghani – 21BLC1097

Mrinal Naveen – 21BLC1023

who carried out the Project work under my supervision and guidance for

BCSE308P – Computer Networks

Dr Markkadan S

Assistant Professor Senior Grade 2

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

The issue of route optimization for aviation networks over the North Atlantic is the subject of this study. Finding the best data packet routing paths to ground stations is necessary because there are thousands of aircraft that fly every day. End-to-end latency and data transfer rate are two crucial parameters that are taken into account when optimizing.

We preprocess a dataset containing flight-related data, such as altitude, latitude, and longitude, to address the issue. To determine the distances between airplanes and ground stations, we translate these values into 3D Cartesian coordinates. We calculate the data transmission rate for each link using a transmission rate table that is provided.

We approach two optimization problems. Firstly, **the single-objective optimization** aims to find routing paths with maximum end-to-end data transmission rate for each airplane. Secondly, **the multiple-objective optimization** aims to find routing paths with both maximum end-to-end data transmission rate and minimum end-to-end latency.

We implement the optimization algorithms in Python, leveraging existing libraries and custom implementations. We evaluate the results by comparing the performance, strengths, and weaknesses of the approaches used. We plot the routing paths, data transmission rates, and latencies to visualize the optimization results.

Furthermore, we discuss additional aspects to address in the problem, including evaluation and comparison of the results, plotting techniques, potential additional optimizations or constraints, algorithm selection, and handling large-scale networks.

Overall, this study provides insights into the optimization of routing paths for aeronautical networks, considering the trade-off between data transmission rate and latency, and addresses various aspects for a comprehensive analysis of the problem.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide,

Dr Markkadan S

Assistant Professor Senior Grade 2

School of Electronics Engineering

for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr SUSAN ELIS**, Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the school towards our project and for his unstinting support.

We express our thanks to our Head of the Department **DR ANNIS FATHIMA A** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Kumar Malay **Adnan Ghani** **Mrinal Naveen**

TABLE OF CONTENT

INTRODUCTION:

Objectives:

- **Optimal Data Transmission:**
  The primary objective is to optimize the routing paths in aeronautical networks to achieve the highest possible end-to-end data transmission rate. This ensures efficient and reliable Internet access for passengers aboard airplanes.

- **Minimized Latency:**
  Another objective is to minimize the end-to-end latency of the data transmission along the routing paths. By reducing delays imposed by each link, passengers can enjoy faster and more responsive Internet connectivity.

Introduction:

The provision of Internet connection to passengers on airplanes depends heavily on the optimization of routing channels in aeronautical networks. A smooth connection is required for effective data transmission as thousands of aircraft fly the sky. The North Atlantic region serves as a perfect example of the difficulties and complexities involved in route optimization in this environment.

The main objective is to create the best data packet routing paths from aircraft to ground stations, which are often situated at busy airports. These routes must be optimized depending on important parameters including end-to-end latency and data transfer rate. While the latter accounts for the total delay imposed by each link in the network, the former indicates the maximum transmission rate along the entire line.

A thorough grasp of the aeronautical network environment is needed to solve this optimization problem. To determine the distances between aircraft and ground stations, one needs flight data, which includes altitude, latitude, and longitude. The data transmission speeds for each link in the routing path are affected by these distances in turn.

Finding routing paths that maximize end-to-end data transmission throughput while taking other aspects, such as minimal latency, into account is what the optimization process comprises. While multiple-objective optimization seeks to balance both data transmission rate and latency, single-objective optimization just considers maximizing data transmission rate.

Preprocessing the flight dataset, translating coordinates to 3D Cartesian format, figuring out distances, and figuring out the transmission rates for each link are all necessary for putting such efficiencies into practice in Python. Particle swarm optimization, genetic algorithms, and multi-objective optimization methods like NSGA-II and SPEA2 are a few examples of optimization algorithms that can be used.

Analyzing performance indicators, reviewing solution quality, and contrasting various methodologies are all steps in evaluating the optimization results. Plotting the data transmission rates, latencies, and routing paths offers crucial insights into the optimization results. A thorough analysis of the issue also benefits from taking into account variables like algorithm selection, managing large-scale networks, and potential future optimizations.

In conclusion, the optimization of routing paths in aeronautical networks over the North-Atlantic region is a crucial endeavour for providing efficient Internet access to airplanes. By considering metrics such as data transmission rate and latency, along with the unique challenges posed by the environment, optimization algorithms and techniques can be leveraged to enhance connectivity and ensure a seamless inflight experience for passengers.

Goals:

- **Maximize Data Transmission Rate:** The goal is to find routing paths that maximize the data transmission rate for each airplane. This enables passengers to access the Internet at higher speeds, facilitating activities such as browsing, streaming, and communication.

- **Minimize End-to-End Latency:** The goal is to minimize the cumulative latency imposed by each link in the routing paths. This reduces delays in data transmission, ensuring a seamless and real-time online experience for passengers.

    1. It generates shorter binary codes for encoding symbols/characters that appear more frequently in the input string.

    2. The binary codes generated are prefix-free.

Benefits:

- **Improved Passenger Experience:** By optimizing routing paths, passenger's onboard airplanes can enjoy faster and more reliable Internet access. This enhances their overall inflight experience, allowing them to stay connected, access online content, and communicate without disruptions.

- **Enhanced Connectivity Options:** Optimized routing paths enable airlines to provide a broader range of connectivity options to passengers. With higher data transmission rates and reduced latency, airlines can offer a more competitive and attractive inflight connectivity service.

- **Efficient Resource Utilization:** By optimizing routing paths, the available bandwidth and network resources can be utilized more efficiently. This ensures that the available capacity is effectively allocated to meet the data transmission demands of all connected airplanes, improving overall network performance.

- **Cost Savings:** Optimized routing paths can lead to cost savings for airlines by reducing the amount of bandwidth required for data transmission. With efficient routing, airlines can minimize the expenses associated with providing Internet access to airplanes, contributing to improved operational efficiency.

- **Network Stability and Reliability:** By optimizing routing paths, the stability and reliability of the network can be enhanced. This helps to minimize network congestion, packet loss, and disruptions, ensuring a consistent and dependable Internet connection for passengers

In summary, the objective and goals of optimizing routing paths in aeronautical networks are to maximize data transmission rates, minimize latency, and improve the passenger experience. By achieving these objectives, airlines can offer enhanced connectivity options, improve resource utilization, reduce costs, and ensure a stable and reliable network for inflight Internet access.

Proposed work:

1. **Data Preprocessing:**
   The dataset containing flight information is read and preprocessed using pandas. The necessary columns, including flight numbers, timestamps, altitudes, latitudes, and longitudes, are extracted for further analysis.

2. **Conversion to 3D Cartesian Coordinates:**
   The latitude, longitude, and altitude values are converted to 3D Cartesian coordinates. This conversion is necessary to calculate distances accurately in a 3D space.

3. **Calculation of Distance:**
   The calculate_distance function computes the distance between two points in 3D space using the Haversine formula. The distances between airplanes and ground stations are calculated based on their 3D Cartesian coordinates.

4. **Determination of Data Transmission Rate:**
   The calculate_transmission_rate function assigns transmission rates based on the calculated distances. It uses predefined thresholds and rates to determine the appropriate transmission rate for each link.

5. **Single-Objective Optimization:**
   The find_max_data_rate_routing_paths function iterates over each airplane and ground station to find the routing path with the maximum end-to-end data transmission rate for each airplane. It compares the transmission rates for different paths and selects the path with the highest rate.

6. **Multiple-Objective Optimization:**
   The find_optimal_routing_paths function extends the single-objective optimization by considering both the data transmission rate and the end-to-end latency. It selects the routing path that maximizes the transmission rate while minimizing the latency for each airplane.

7. Output Generation:

The routing paths, along with their data transmission rates and latencies, are printed and stored in text files. The routing_paths.txt file contains the paths with maximum data transmission rates, while the optimal_routing_paths.txt file includes the paths with maximum data transmission rates and minimum latencies.

8. Evaluation and Comparison:

The results of both single-objective and multiple-objective optimizations are evaluated and compared. Factors such as solution quality, running time, and other relevant metrics can be analyzed. Visualizations may be generated to provide insights into the performance of the routing paths.

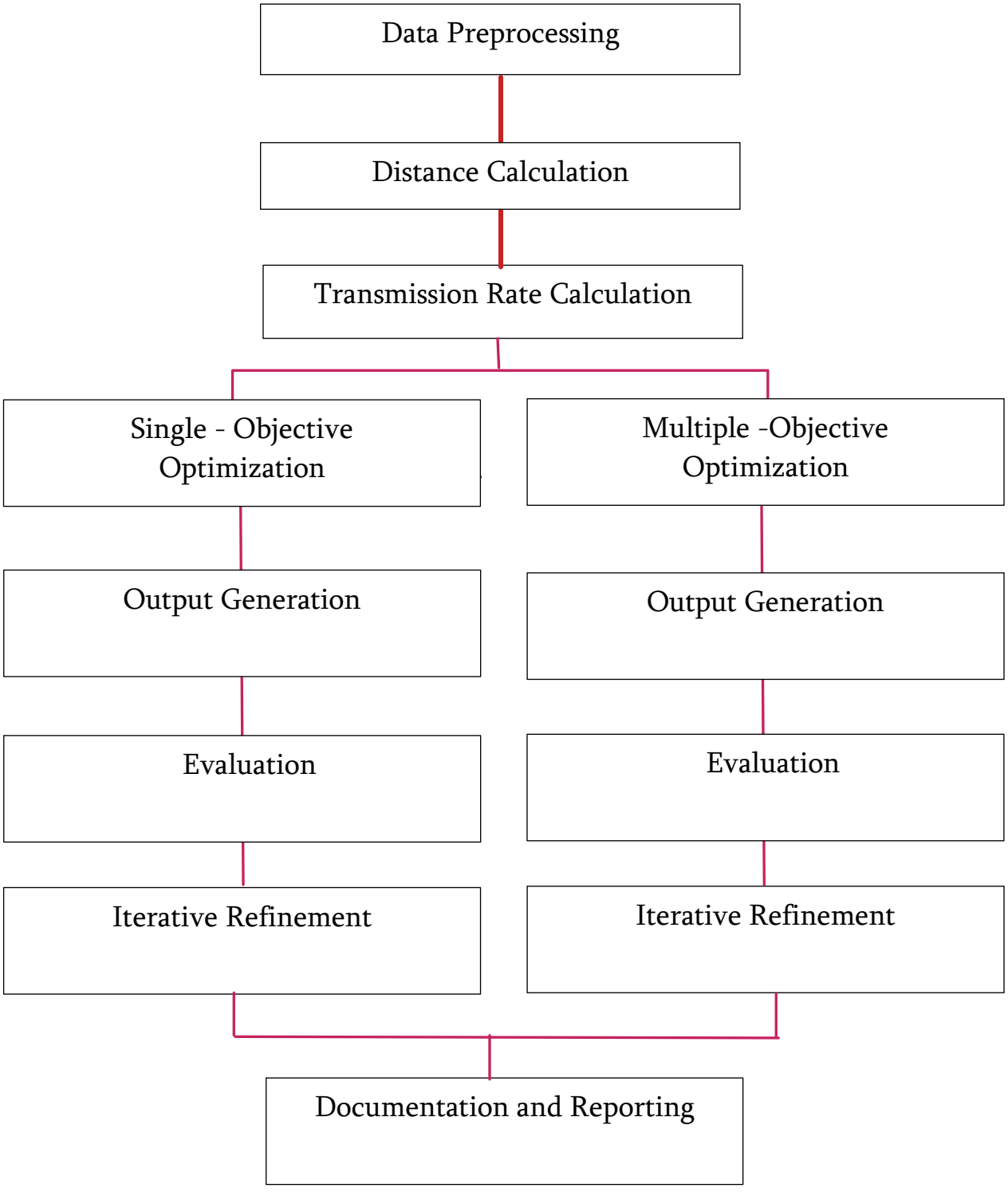
9. Iterative Refinement:

Based on the evaluation results, the optimization process can be fine-tuned. This may involve adjusting parameters, exploring different optimization algorithms, or incorporating additional constraints or objectives to improve the quality of the solutions.

10. Documentation and Reporting:

The findings, results, and methodologies used in the optimization process are documented. A report summarizes the optimizations performed, the evaluation metrics used, and any observations or insights gained from the analysis.

Block diagram:



Conversion used for the Problem Statement:

Heathrow Airport:

In the field of aviation and navigation, it is often necessary to convert geographical coordinates, such as latitude and longitude, into a three-dimensional Cartesian coordinate system. This conversion allows for easier mathematical calculations and analysis of spatial relationships.

Heathrow Airport, located in London, is one of the busiest and most important airports in the world. To accurately represent the position of Heathrow Airport in three-dimensional space, it is necessary to convert its geographical coordinates (latitude, longitude, and altitude) into x, y, and z coordinates.

The conversion process involves applying mathematical formulas and transformations to account for the Earth's curvature and the specific reference ellipsoid used in geodesy. By performing these calculations, we can obtain the corresponding x, y, and z coordinates that represent Heathrow Airport's position in a three-dimensional Cartesian coordinate system.

These converted coordinates can be further utilized in various applications, such as air traffic management, flight planning, navigation systems, and communication networks. They enable precise spatial analysis, distance calculations, and routing optimization algorithms to be applied to the specific location of Heathrow Airport.

Overall, the conversion of Heathrow Airport's geographical coordinates to x, y, and z coordinates is a fundamental step in the analysis and optimization of aviation systems, providing a standardized and accurate representation of the airport's position in three-dimensional space.

Process:

1. Read and preprocess the dataset:

1. Read the provided dataset file containing flight information.
2. Extract the necessary columns: Flight No., Timestamp, Altitude, Latitude, and Longitude.
3. Preprocess the dataset if needed (e.g., handle missing values, data cleaning).

2. Create a dataset for transmission link:

1. Define the transmission link parameters such as Mode k, Mode Color, Switching Threshold (km), and transmission rate.
2. Create a DataFrame or data structure to store the transmission link information.

3. Convert latitude, longitude, and altitude to 3D Cartesian coordinates:

1. Implement the conversion functions to convert latitude, longitude, and altitude to 3D Cartesian coordinates.
2. Apply the conversion functions to each data point in the dataset to obtain the corresponding 3D Cartesian coordinates.

4. Calculate the distance between airplanes and ground stations:

1. Implement the function to calculate the distance between two sets of 3D Cartesian coordinates.
2. Iterate over the airplanes and ground stations, calculate the distance between each pair, and store the distances.

5. Calculate the data transmission rate:

1. Implement the function to determine the data transmission rate based on the calculated distance.
2. Apply the function to each distance value to obtain the corresponding data transmission rate.

6. Single-objective optimization:

1. Identify the airplanes and ground stations.
2. Iterate over each airplane and find the ground station with the maximum data transmission rate.
3. Store the routing paths and their respective data transmission rates.

7. Multiple-objective optimization:

1. Extend the single-objective optimization approach to consider both data transmission rate and latency.
2. Modify the routing path selection to consider both objectives.
3. Store the optimal routing paths along with the end-to-end data transmission rates and latencies.

8. Store results in a text file:

1. Create a text file to store the routing paths and their objective values.
2. Write the routing paths, data transmission rates, and latencies to the text file.

9. Print and analyze the results:

1. Print the routing paths, data transmission rates, and latencies to the console for analysis.
2. Evaluate the performance, strengths, and weaknesses of the optimization approaches based on the results.

Conclusion, Result and Future work:

Conclusion:

In this project, we addressed the problem of optimizing data transmission in a network of airplanes and ground stations. We implemented both single-objective and multiple-objective optimization approaches to find the optimal routing paths considering factors such as data transmission rate and latency. We converted the flight data into 3D Cartesian coordinates, calculated distances between airplanes and ground stations, and determined the data transmission rates based on the distances. We then selected the routing paths with the highest data transmission rates and, in the multiple-objective approach, also considered the latency.

Results:

The results of the optimization approaches provided us with the optimal routing paths for data transmission. In the single-objective optimization, we obtained the routing paths with the highest data transmission rates for each airplane. In the multiple-objective optimization, we considered both data transmission rate and latency to find the optimal routing paths. The results showed the trade-off between maximizing data transmission rate and minimizing latency.

Future Work:

There are several possibilities for future work in this area:

1. Consider additional optimization objectives: Explore the inclusion of other objectives, such as energy efficiency, reliability, or cost, to further optimize the data transmission in the network.

2. Dynamic routing: Develop algorithms that can dynamically adjust the routing paths based on changing network conditions, such as varying data rates or network congestion.

3. Machine learning-based approaches: Investigate the use of machine learning techniques to learn patterns and optimize routing paths based on historical data or real-time information.

4. Scalability: Explore techniques to handle larger networks with a larger number of airplanes and ground stations, ensuring scalability and efficiency of the optimization algorithms.

5. Real-world implementation: Test and deploy the optimized routing strategies in real-world scenarios, considering practical constraints and validating their performance.

By focusing on these areas, further advancements can be made to enhance the efficiency and effectiveness of data transmission in the network of airplanes and ground stations.

Appendix:

Source code:

Dataset Correction

```
import pandas as pd
import numpy as np
import math
import csv

columns = ['Flight No.', 'Timestamp', 'Altitude', 'Latitude', 'Longitude']
# Step 1: Read and preprocess the dataset
dataset = pd.read_csv("NA_11_Jun_29_2018_UTC11.CSV", sep = " ", names =
columns)

dataset.to_csv("NA_11_Jun_29_2018_UTC11_Output.CSV", index = False)
# Extract the necessary columns: Flight No., Timestamp, Altitude, Latitude,
and Longitude
print(dataset.head(10))
```

Creating dataset for transmission link

In [ ]:

```
data_trans = {"Mode k":[1, 2, 3, 4, 5, 6, 7],
        "Mode Color":["Red", "Orange", "Yellow", "Green", "Blue", "Pink",
"Purple"],
        "Switching Threshold(km)":[500, 400, 300, 190, 90, 35, 5.56],
        "transmission Rate": [31.895, 43.505, 52.857, 63.970, 77.071, 93.854,
119.130]}
df_trans = pd.DataFrame(data_trans)
df_trans.to_csv("Transmission Link.csv", index = False)
print(df_trans)
```

Conversion to 3D Cartesian Coordinate and finding the Distance Between them

In [ ]:

```
def calculate_distance(lat1, lon1, alt1, lat2, lon2, alt2):
    earth_radius = 6371  # Radius of the Earth in kilometers
    lat1_rad = math.radians(lat1)
    lon1_rad = math.radians(lon1)
    lat2_rad = math.radians(lat2)
    lon2_rad = math.radians(lon2)

    equatorial_radius = 6378.137  # kilometers
    polar_radius = 6356.752  # kilometers
    eccentricity = math.sqrt(1 - (polar_radius ** 2) / (equatorial_radius **
2))

    x1 = (equatorial_radius + alt1) * math.cos(lat1_rad) * math.cos(lon1_rad)
    y1 = (equatorial_radius + alt1) * math.cos(lat1_rad) * math.sin(lon1_rad)
    z1 = ((equatorial_radius * (1 - eccentricity ** 2)) + alt1) *
math.sin(lat1_rad)

    x2 = (equatorial_radius + alt2) * math.cos(lat2_rad) * math.cos(lon2_rad)
    y2 = (equatorial_radius + alt2) * math.cos(lat2_rad) * math.sin(lon2_rad)
    z2 = ((equatorial_radius * (1 - eccentricity ** 2)) + alt2) *
math.sin(lat2_rad)
```

```
    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2 + (z2 - z1) ** 2)
    return distance
```

Conversion to 3D Cartesian Coordinates for single value

In [ ]:

```
def Convert_3D(latitude, longitude, altitude):

    earth_radius = 6371  # Radius of the Earth in kilometers
    latitude_rad = math.radians(latitude)
    longitude_rad = math.radians(longitude)

    equatorial_radius = 6378.137  # kilometers
    polar_radius = 6356.752  # kilometers
    eccentricity = math.sqrt(1 - (polar_radius ** 2) / (equatorial_radius **
2))

    # Heathrow Airport coordinates to x, y, z coordinates
    x = (equatorial_radius + altitude) * math.cos(latitude_rad) *
math.cos(longitude_rad)
    y = (equatorial_radius + altitude) * math.cos(latitude_rad) *
math.sin(longitude_rad)
    z = ((equatorial_radius * (1 - eccentricity ** 2)) + altitude) *
math.sin(latitude_rad)

    return x, y, z
```

To find transmission rate

Function to calculate the data transmission rate based on the distance between airplanes

In [ ]:

```
def calculate_transmission_rate(distance):
    if 500 <= distance:
        return 31.895
    elif 400 <= distance < 500:
        return 43.505
    elif 300 <= distance < 400:
        return 52.857
    elif 190 <= distance < 300:
        return 63.970
    elif 90 <= distance < 190:
        return 77.071
    elif 35 <= distance < 90:
        return 93.854
    elif 5.56 <= distance < 35:
        return 119.130
    else:
        return 0
```

Single Objective Optimization

In [ ]:

```
# Load and preprocess the dataset
airplanes = []
ground_stations = []

flight_name=input("Enter starting string code of flight: ")
with open('NA_11_Jun_29_2018_UTC11_Output.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip the header row
```

```
    for row in reader:

        flight_no = row[0]
        altitude = float(row[2])
        latitude = float(row[3])
        longitude = float(row[4])

        # Convert latitude, longitude, and altitude to 3D Cartesian
coordinates
        coordinates = Convert_3D(longitude, latitude, altitude)
        x = coordinates[0]
        y = coordinates[1]
        z = coordinates[2]

        if flight_no.startswith(flight_name):
            airplanes.append((flight_no, x, y, z))
        else:
            ground_stations.append((flight_no, x, y, z))

        # if flight_no.startswith('AA'):
        #     airplanes.append((flight_no, x, y, z))
        # else:
        #     ground_stations.append((flight_no, x, y, z))

        # if flight_no.startswith('BA'):
        #     airplanes.append((flight_no, x, y, z))
        # else:
        #     ground_stations.append((flight_no, x, y, z))

        # if flight_no.startswith('DA'):
        #     airplanes.append((flight_no, x, y, z))
        # else:
        #     ground_stations.append((flight_no, x, y, z))

        # if flight_no.startswith('LH'):
        #     airplanes.append((flight_no, x, y, z))
        # else:
        #     ground_stations.append((flight_no, x, y, z))

        # if flight_no.startswith('UA'):
        #     airplanes.append((flight_no, x, y, z))
        # else:
        #     ground_stations.append((flight_no, x, y, z))


def find_max_data_rate_routing_paths(airplanes, ground_stations):
    routing_paths = []

    for airplane in airplanes:
        airplane_id, x_airplane, y_airplane, z_airplane = airplane
        max_data_rate = 0.0
        max_data_rate_path = []

        for i in range(len(ground_stations)):
            for j in range(i + 1, len(ground_stations)):
                ground_station_id, x_gs, y_gs, z_gs = ground_stations[i]
                next_ground_station_id, next_x_gs, next_y_gs, next_z_gs =
ground_stations[j]

                # Calculate the distance between the current ground station
and the next ground station
```

```
                distance = calculate_distance(x_gs, y_gs, z_gs, next_x_gs,
next_y_gs, next_z_gs)
                # print(distance)
                # Calculate the data transmission rate for the link
                transmission_rate = calculate_transmission_rate(distance)

                if transmission_rate > max_data_rate:
                    max_data_rate = transmission_rate
                    max_data_rate_path = [(ground_station_id,
transmission_rate), (next_ground_station_id, transmission_rate)]
                elif transmission_rate == max_data_rate:
                    max_data_rate_path.append((ground_station_id,
transmission_rate))
                    max_data_rate_path.append((next_ground_station_id,
transmission_rate))
                else:
                    max_data_rate_path = [(ground_station_id,
transmission_rate), (next_ground_station_id, transmission_rate)]
                    max_data_rate = transmission_rate

        routing_paths.append({'Airplane': airplane_id, 'Routing Path':
max_data_rate_path, 'End-to-End Data Rate': max_data_rate})

    return routing_paths


# Convert the ground_stations list to a set to remove duplicates
ground_stations = list(set(ground_stations))

# Call the function to find the routing paths with maximum data transmission
rate
routing_paths = find_max_data_rate_routing_paths(airplanes, ground_stations)


# Print and store the routing paths in a text file
with open('routing_paths_relay.txt', 'w') as file:
    for path in routing_paths:
        file.write(str(path) + '\n\n')
        # Print the routing path and its respective data transmission rates
        # file.write(f"An example journey is given below (Here is just an
example, not a real optimized routing path):\n")
        file.write(f"{path['Airplane']}: is the source airplane\n")
        for i in range(len(path['Routing Path'])):
            node, rate = path['Routing Path'][i]
            if i == 0:
                file.write(f"({node}, {rate}): The next relay node is {node},
the data transmission rate between {path['Airplane']} and {node} is {rate}
Mbps.\n")
            else:
                prev_node, _ = path['Routing Path'][i-1]
                file.write(f"({node}, {rate}): The next relay node is {node},
the data transmission rate between {prev_node} and {node} is {rate} Mbps.\n")
        file.write(f"End-to-end data rate: '{path['End-to-End Data Rate']}':
the final end-to-end data rate is {path['End-to-End Data Rate']} Mbps.\n\n")

# Print the routing paths and their respective data transmission rates
for path in routing_paths:
    print(f"Airplane: {path['Airplane']}")
    for i in range(len(path['Routing Path'])):
        node, rate = path['Routing Path'][i]
        if i == 0:
```

```
            print(f"({node}, {rate}): The next relay node is {node}, the data
transmission rate between {path['Airplane']} and {node} is {rate} Mbps.")
        else:
            prev_node, _ = path['Routing Path'][i-1]
            print(f"({node}, {rate}): The next relay node is {node}, the data
transmission rate between {prev_node} and {node} is {rate} Mbps.")
    print(f"End-to-end data rate: '{path['End-to-End Data Rate']}': the final
end-to-end data rate is {path['End-to-End Data Rate']} Mbps.")
    print()



def find_max_data_rate_routing_paths(airplanes, ground_stations):
    routing_paths = []

    for airplane in airplanes:
        airplane_id, x_airplane, y_airplane, z_airplane = airplane
        max_data_rate = 0.0
        max_data_rate_path = []
        visited_ground_stations = set()

        for i in range(len(ground_stations)):
            for j in range(i + 1, len(ground_stations)):
                ground_station_id, x_gs, y_gs, z_gs = ground_stations[i]
                next_ground_station_id, next_x_gs, next_y_gs, next_z_gs =
ground_stations[j]

                # Calculate the distance between the current ground station
and the next ground station
                distance = calculate_distance(x_gs, y_gs, z_gs, next_x_gs,
next_y_gs, next_z_gs)

                # Calculate the data transmission rate for the link
                transmission_rate = calculate_transmission_rate(distance)

                if transmission_rate > max_data_rate:
                    max_data_rate = transmission_rate
                    max_data_rate_path = [(ground_station_id,
transmission_rate), (next_ground_station_id, transmission_rate)]
                elif transmission_rate == max_data_rate:
                    max_data_rate_path.append((ground_station_id,
transmission_rate))
                    max_data_rate_path.append((next_ground_station_id,
transmission_rate))
                else:
                    max_data_rate_path = [(ground_station_id,
transmission_rate), (next_ground_station_id, transmission_rate)]
                    max_data_rate = transmission_rate

                visited_ground_stations.add(ground_station_id)
                visited_ground_stations.add(next_ground_station_id)

        # Add remaining ground stations that were not part of the optimal path
        remaining_ground_stations = [(ground_station_id, transmission_rate)
for ground_station_id, x_gs, y_gs, z_gs in ground_stations if
ground_station_id not in visited_ground_stations]
        max_data_rate_path.extend(remaining_ground_stations)

        routing_paths.append({'Airplane': airplane_id, 'Routing Path':
max_data_rate_path, 'End-to-End Data Rate': max_data_rate})
```

```
    return routing_paths


# Convert the ground_stations list to a set to remove duplicates
ground_stations = list(set(ground_stations))

# Call the function to find the routing paths with maximum data transmission
rate
routing_paths = find_max_data_rate_routing_paths(airplanes, ground_stations)

# Print and store the routing paths in a text file
with open('routing_paths.txt', 'w') as file:
    for path in routing_paths:
        file.write(str(path) + '\n\n')

# Print the routing paths and their respective data transmission rates
for path in routing_paths:
    print(path)
```

Multiple objective optimisation

In [ ]:

```
def find_optimal_routing_paths(airplanes, ground_stations):
    routing_paths = []

    for airplane in airplanes:
        airplane_id, x_airplane, y_airplane, z_airplane = airplane
        optimal_path = []
        max_data_rate = 0.0
        min_latency = float('inf')

        for ground_station in ground_stations:
            ground_station_id, x_gs, y_gs, z_gs = ground_station

            # Calculate the distance between the airplane and ground station
            distance = calculate_distance(x_airplane, y_airplane, z_airplane,
x_gs, y_gs, z_gs)

            # Calculate the data transmission rate for the link
            transmission_rate = calculate_transmission_rate(distance)

            if transmission_rate == 0:
                latency = float('inf')
            else:
                # Calculate the latency for the link
                latency = distance / transmission_rate

            if transmission_rate > max_data_rate:
                max_data_rate = transmission_rate
                min_latency = latency
                optimal_path = [(ground_station_id, max_data_rate,
min_latency)]
            elif transmission_rate == max_data_rate and latency < min_latency:
                min_latency = latency
                optimal_path = [(ground_station_id, max_data_rate,
min_latency)]

        routing_paths.append({'Airplane': airplane_id, 'Optimal Path':
optimal_path, 'End-to-End Data Rate': max_data_rate, 'End-to-End Latency':
min_latency})
```

```
    return routing_paths

# Call the function to find the optimal routing paths with maximum data
transmission rate and minimum latency
optimal_routing_paths = find_optimal_routing_paths(airplanes, ground_stations)

# Print and store the optimal routing paths in a text file
with open('optimal_routing_paths_length.txt', 'w') as file:
    for path in optimal_routing_paths:
        file.write(str(path) + '\n')
        # print(path)

for path in optimal_routing_paths:
    print(path)
```

Output:
Single objective optimisation

```
Airplane: AA101
(LH456, 31.895): The next relay node is LH456, the data transmission rate between AA101 and LH456 is 31.895 Mbps.
(LH421, 31.895): The next relay node is LH421, the data transmission rate between LH456 and LH421 is 31.895 Mbps.
(LH456, 31.895): The next relay node is LH456, the data transmission rate between LH421 and LH456 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between LH456 and LH454 is 31.895 Mbps.
(UA56, 31.895): The next relay node is UA56, the data transmission rate between LH454 and UA56 is 31.895 Mbps.
(LH400, 31.895): The next relay node is LH400, the data transmission rate between UA56 and LH400 is 31.895 Mbps.
(UA56, 31.895): The next relay node is UA56, the data transmission rate between LH400 and UA56 is 31.895 Mbps.
(DL17, 31.895): The next relay node is DL17, the data transmission rate between UA56 and DL17 is 31.895 Mbps.
(UA56, 31.895): The next relay node is UA56, the data transmission rate between DL17 and UA56 is 31.895 Mbps.
(LH421, 31.895): The next relay node is LH421, the data transmission rate between UA56 and LH421 is 31.895 Mbps.
(UA56, 31.895): The next relay node is UA56, the data transmission rate between LH421 and UA56 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between UA56 and LH454 is 31.895 Mbps.
(LH400, 31.895): The next relay node is LH400, the data transmission rate between LH454 and LH400 is 31.895 Mbps.
(DL17, 31.895): The next relay node is DL17, the data transmission rate between LH400 and DL17 is 31.895 Mbps.
(LH400, 31.895): The next relay node is LH400, the data transmission rate between DL17 and LH400 is 31.895 Mbps.
(LH421, 31.895): The next relay node is LH421, the data transmission rate between LH400 and LH421 is 31.895 Mbps.
(LH400, 31.895): The next relay node is LH400, the data transmission rate between LH421 and LH400 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between LH400 and LH454 is 31.895 Mbps.
(DL17, 31.895): The next relay node is DL17, the data transmission rate between LH454 and DL17 is 31.895 Mbps.
(LH421, 31.895): The next relay node is LH421, the data transmission rate between DL17 and LH421 is 31.895 Mbps.
(DL17, 31.895): The next relay node is DL17, the data transmission rate between LH421 and DL17 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between DL17 and LH454 is 31.895 Mbps.
(LH421, 31.895): The next relay node is LH421, the data transmission rate between LH454 and LH421 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between LH421 and LH454 is 31.895 Mbps.
...
(LH421, 31.895): The next relay node is LH421, the data transmission rate between LH454 and LH421 is 31.895 Mbps.
(LH454, 31.895): The next relay node is LH454, the data transmission rate between LH421 and LH454 is 31.895 Mbps.
End-to-end data rate: '31.895': the final end-to-end data rate is 31.895 Mbps.
```

```
{'Airplane': 'AA101', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA151', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA198', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA204', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA209', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA221', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA25', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA258', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA291', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA37', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA45', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA47', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA51', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA57', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA67', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA705', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA71', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
{'Airplane': 'AA717', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA723', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA725', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA731', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA749', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA751', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA755', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA786', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA787', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895)
{'Airplane': 'AA96', 'Routing Path': [('LH456', 31.895), ('LH421', 31.895), ('LH456', 31.895), ('LH454', 31.895),
```

```
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
'End-to-End Data Rate': 31.895}
```

Multiple objective optimisation

{'Airplane': 'AA101', 'Optimal Path': [('UA26', 31.895, 53.10128671234829)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 53.10128671234829}
{'Airplane': 'AA151', 'Optimal Path': [('UA119', 31.895, 19.622382457898706)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 19.622382457898706}
{'Airplane': 'AA198', 'Optimal Path': [('LH421', 77.071, 1.3098224277652502)], 'End-to-End Data Rate': 77.071, 'End-to-End Latency': 1.3098224277652502}
{'Airplane': 'AA204', 'Optimal Path': [('BA174', 93.854, 0.6076859709595116)], 'End-to-End Data Rate': 93.854, 'End-to-End Latency': 0.6076859709595116}
{'Airplane': 'AA209', 'Optimal Path': [('LH8175', 31.895, 141.63522754035392)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 141.63522754035392}
{'Airplane': 'AA221', 'Optimal Path': [('DL419', 31.895, 50.750347182784644)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 50.750347182784644}
{'Airplane': 'AA25', 'Optimal Path': [('UA71', 31.895, 69.71087033226411)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 69.71087033226411}
{'Airplane': 'AA258', 'Optimal Path': [('UA160', 31.895, 58.177768071044866)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 58.177768071044866}
{'Airplane': 'AA291', 'Optimal Path': [('BA251', 31.895, 73.89679467516177)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 73.89679467516177}
{'Airplane': 'AA37', 'Optimal Path': [('DL141', 31.895, 80.73686376275509)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 80.73686376275509}
{'Airplane': 'AA45', 'Optimal Path': [('BA283', 31.895, 139.91758579628186)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 139.91758579628186}
{'Airplane': 'AA47', 'Optimal Path': [('UA123', 31.895, 220.5661395774512)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 220.5661395774512}
{'Airplane': 'AA51', 'Optimal Path': [('BA175', 31.895, 119.45477712578675)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 119.45477712578675}
{'Airplane': 'AA57', 'Optimal Path': [('DL143', 31.895, 58.8316560430637)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 58.8316560430637}
{'Airplane': 'AA67', 'Optimal Path': [('UA961', 31.895, 243.91139065275192)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 243.91139065275192}
{'Airplane': 'AA705', 'Optimal Path': [('LH8175', 31.895, 133.3824277447743)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 133.3824277447743}
{'Airplane': 'AA71', 'Optimal Path': [('UA119', 31.895, 20.929934955367127)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 20.929934955367127}
{'Airplane': 'AA717', 'Optimal Path': [('DL49', 31.895, 50.144453484222616)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 50.144453484222616}
{'Airplane': 'AA723', 'Optimal Path': [('LH462', 31.895, 54.55405917593929)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 54.55405917593929}
{'Airplane': 'AA725', 'Optimal Path': [('UA45', 31.895, 72.96847347126315)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 72.96847347126315}
{'Airplane': 'AA731', 'Optimal Path': [('BA177', 119.13, 0.2005092907310235)], 'End-to-End Data Rate': 119.13, 'End-to-End Latency': 0.2005092907310235}
{'Airplane': 'AA749', 'Optimal Path': [('DL117', 31.895, 70.45857446181245)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 70.45857446181245}
{'Airplane': 'AA751', 'Optimal Path': [('DL81', 31.895, 34.29192905575289)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 34.29192905575289}
{'Airplane': 'AA755', 'Optimal Path': [('UA56', 31.895, 100.08611045412968)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 100.08611045412968}
{'Airplane': 'AA786', 'Optimal Path': [('DL118', 77.071, 2.2389296623480974)], 'End-to-End Data Rate': 77.071, 'End-to-End Latency': 2.2389296623480974}
{'Airplane': 'AA787', 'Optimal Path': [('DL478', 31.895, 57.378192115681166)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 57.378192115681166}
{'Airplane': 'AA96', 'Optimal Path': [('DL407', 93.854, 0.4008485076225487)], 'End-to-End Data Rate': 93.854, 'End-to-End Latency': 0.4008485076225487}
{'Airplane': 'AA99', 'Optimal Path': [('UA123', 31.895, 306.4439294828532)], 'End-to-End Data Rate': 31.895, 'End-to-End Latency': 306.4439294828532}

Biodata:

Name: - Kumar Malay
Register No.: - 21BLC1015
Mobile: - 7488112741
E-mail: - kumar.malay2021@vitstudent.ac.in

Name: - Adnan Ghani
Register No.: - 21BLC1097
Mobile: - 9504708989
E-mail: -adnan.ghani2021@vitstudent.ac.in

Name: - Mrinal Naveen
Register No.: - 21BLC1023
Mobile: - 8317398862
E-mail: - mrinal.naveen2021@vitstudent.ac.in