OPTIMIZATION OF UITM BUS ROUTING USING ANT COLONY OPTIMIZATION

GROUP MEMBERS:
NUR AZRAWINA BINTI AHMAD KONTAR (2020847224)
HASYA FARWIZAH BINTI ABDUL HALIM (2020621482)
SHARIFAH SYAMIRAH BINTI SYED AHMAD ZULFA (2020878228)
INTAN NURUL ASMARIZA BINTI NOOR AZMAN (2020489974)

TABLE OF CONTENTS

01

INTRODUCTION

What our project is all about

INTRODUCTION

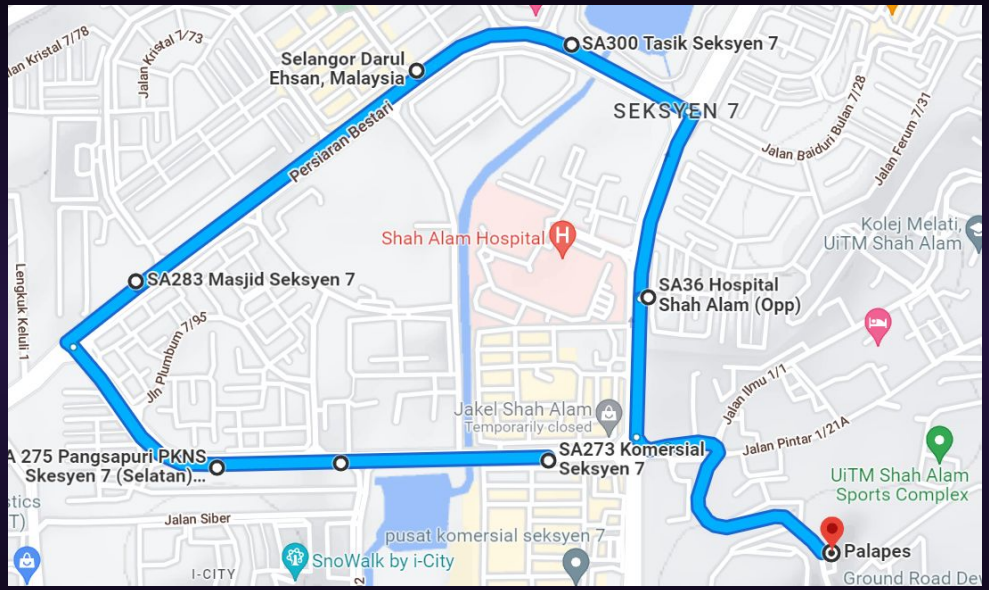
- Campus bus is a vital mode of transportation for students in higher education institutions
- Rising university enrollments have led to issues such as traffic congestion and travel inconvenience on campus

OUR PROJECT

- Determine the most effective and minimal travel distance bus routes for the Universiti Teknologi MARA (UiTM) Shah Alam campus using Ant Colony Optimization
- Focus on bus route C

- Metaheuristic optimization method that is inspired by the foraging behavior of ant colonies
- Ant : Potential solution
- Pheromone trails : the quality of a particular solution
- The algorithm uses pheromone trails left by the ants to guide the search for the best solution
- Application:
  - Vehicle Routing Problem
  - Traveling Salesman Problem
  - Job-Shop Scheduling Problem

Ant Colony Optimization
(ACO)

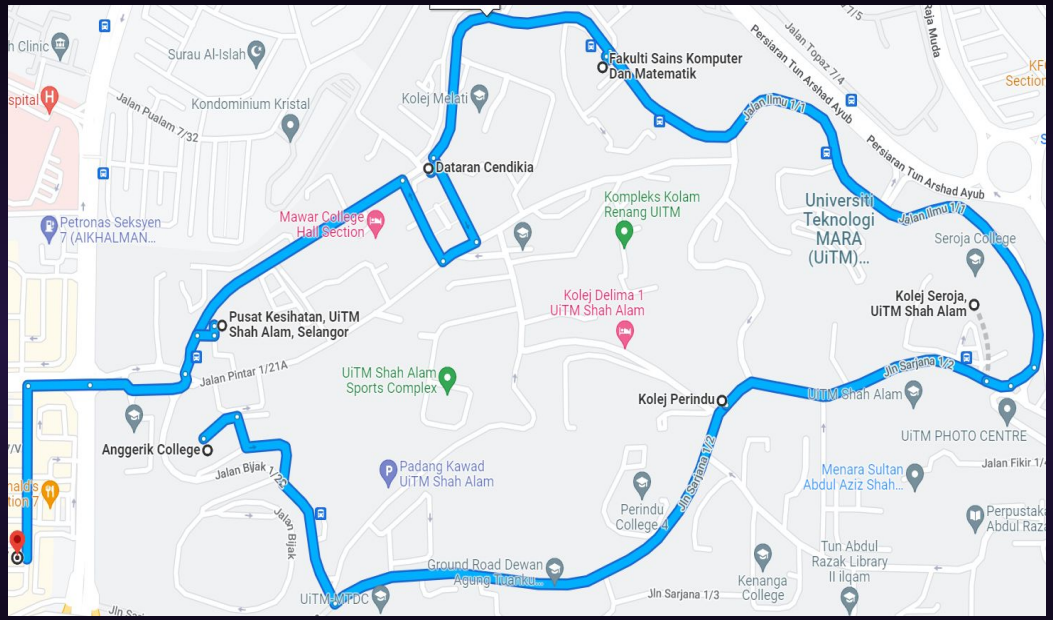


Bus Routing for Bus C near Seksyen 7

Station:

- Hentian Pusat Komersial Seksyen 7 (start)
- Hentian Simpang I-City
- Hentian Flat PKNS
- Hentian Masjid Kristal
- Hentian Polpero
- Hentian Tasik Seksyen 7
- Hentian Hospital
- Hentian Kolej Anggerik

Bus Routing for Bus C in UiTM



Station:

- Hentian Kolej Anggerik (continued)
- Hentian Kolej Perindu
- Hentian Kolej Seroja
- Hentian KPPIM (FSKM)
- Hentian KPPIM 2 (FKM)
- Hentian Dataran Cendekia (DC)
- Hentian Pusat Kesihatan

02

PROBLEM STATEMENT

This section consist of problem statement

PROBLEMS

- Bus visits less priority bus station first as soon as entering the UiTM
- Drag students waiting time

03

PROJECT DESCRIPTION

This section consists of project description

Buses

A total of 6 busses covered Bus C Routing based on their schedule

Bus Station

- A total of 14 bus stations are covered in Bus C Routing
- Distance between the bus stations
- Bus station scattered both on- and off- campus
- Starting point: Hentian Pusat Komersial Seksyen 7

04

SOLUTION REPRESENTATION

Objective Function

Objective: To minimize the distance travelled by the bus during peak hours

Objective function: Pythagoras Theorem
The distance between bus stops:

The shortest overall distance will be calculated to obtain the optimum route for achieving the objective.

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Node	Coordinate of bus stop represented as (x, y)
Ants	Represents a single route
Alpha	Represents the pheromone trail persistence in relation to other factors
Beta	Provides heuristic information in the ants' decision-making process
Rho	Pheromone evaporation rate

Data Acquisition

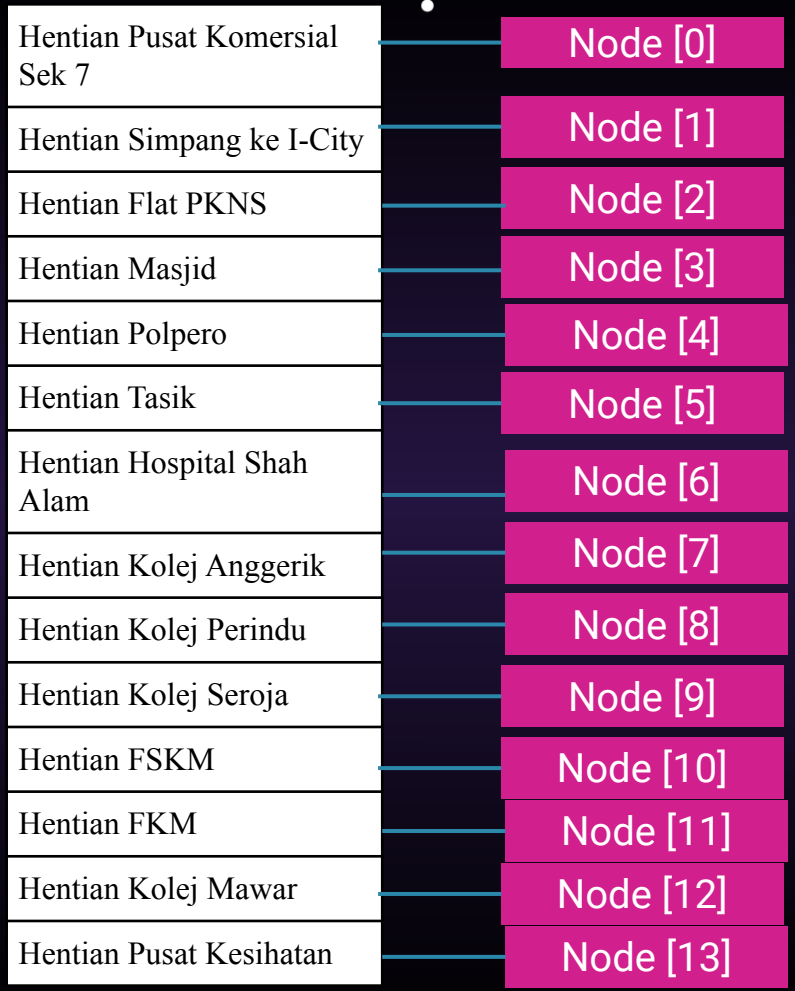
- The data was obtained through UiTM's bus schedule from one of the bus stops
- The stops for Bus C is as follows:

JADUAL PERGERAKAN BAS LALUAN KAMPUS UiTM SHAH ALAM : SEMESTER OKTOBER 2022 - FEBRUARI 2023	
C	**LALUAN 'C' : SEKSYEN 7 - KAMPUS SHAH ALAM**
	(MULA) HENTIAN PUSAT KOMERSIAL SEK. 7 - HENTIAN SIMPANG KE I-CITY - HENTIAN FLAT PKNS - HENTIAN MASJID JALAN KRISTAL - HENTIAN PALPERO - HENTIAN TASIK - HENTIAN HOSPITAL - HENTIAN ANGGERIK - HENTIAN PERINDU - HENTIAN SEROJA - HENTIAN FSKM - HENTIAN FKPM (MASCOM) - HENTIAN MAWAR (DC) - HENTIAN PUSAT KESIHATAN - HENTIAN PUSAT KOMERSIAL SEK. 7 - HENTIAN SIMPANG KE I-CITY - HENTIAN FLAT PKNS - HENTIAN MASJID JALAN KRISTAL **(TAMAT)**.

- Google Maps was used to obtain the exact coordinates for each of the bus stops
- A dataset of coordinates for the bus stops are created consisting of 2 attributes which are longitude and latitude.

Latitude (°)	Latitude (°)
3.070571	101.491649
3.065650	101.495211
3.066866	101.500978

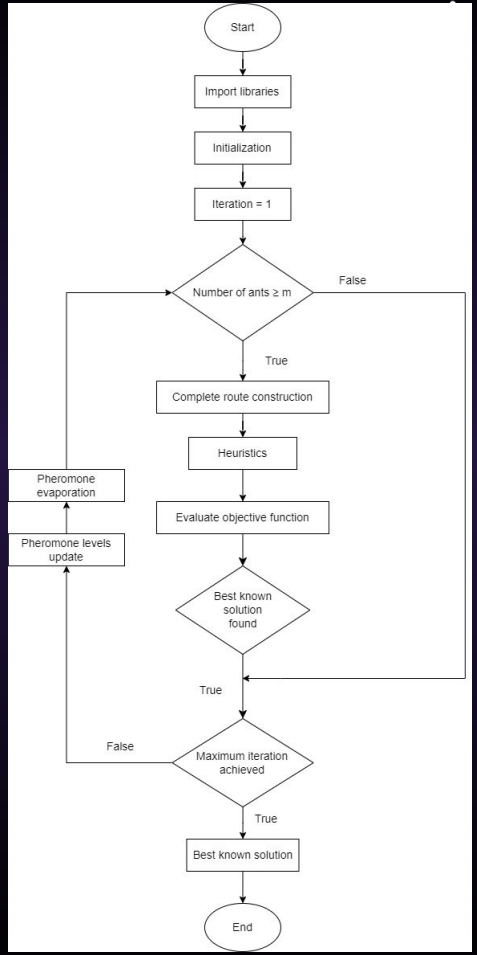
Node Representation



05

ALGORITHM IMPLEMENTATION

FLOWCHART


Start
Import libraries
Initialization
Iteration = 1
Number of ants ≥ m
False
True
Complete route construction
Heuristics
Evaluate objective function
Best known solution found
True
Maximum iteration achieved
False
True
Best known solution
End
Pheromone levels update
Pheromone evaporation

PSEUDOCODE

1. Initialize the pheromone trail for each bus stop.
2. Do
3. For each ant, do
4. Select a random starting location.
5. While not all bus stops have been visited:
6. Select the next location to move based on the pheromone trail and the distance to the next location
7. Update the ant's current location
8. Evaluate the solutions constructed by the ants, calculating the total travel distance
9. Update the pheromone trail based on the quality of the solutions
10. While (stopping condition is true)
11. Return path with smallest/least distance found by the ants as the final solution

1. Import libraries and dataset

Import necessary libraries

```
import math
import random
import matplotlib.pyplot as plt
```

Read contents from .txt file and store in list form

```
with open('busc062.txt','r')as f:
    lines=f.readlines()
    coord_x=[float(line.split()[1]) for line in lines]
    coord_y=[float(line.split()[2]) for line in lines]
```

2. Initialize parameters, define objective function

```
# Step 1. Initialization of parameters
stop_num = len(coord_x)  # the number of stops
dis = [[0 for _ in range(stop_num)] for _ in range(stop_num)]  # distance matrix
for i in range(stop_num):
    for j in range(i, stop_num):
        temp_dis = math.sqrt((coord_x[i] - coord_x[j]) ** 2 + (coord_y[i] - coord_y[j]) ** 2)
        dis[i][j] = temp_dis
        dis[j][i] = temp_dis
pheromone = [[1 for _ in range(stop_num)] for _ in range(stop_num)]
iter_best = []  # the shortest path of each iteration
best_path = []
best_length = 1e6
```

3. Create new path

```python
def construct_path(dis, pheromone, alpha, beta):
    # construct a new path based on distance and pheromone
    path = [0]
    cur_node = 0
    unvisited_stops = [i for i in range(1, len(dis))]
    for i in range(len(dis) - 1):
        roulette_pooling = []
        for stop in unvisited_stops:
            roulette_pooling.append(math.pow(pheromone[cur_node][stop], alpha) * math.pow(1 / dis[cur_node][stop], beta))
        index = roulette(roulette_pooling)
        cur_node = unvisited_stops[index]
        path.append(cur_node)
        unvisited_stops.pop(index)
    path.append(0)
    return path
```

4. Calculate length of path

```python
def cal_dis(dis, path):
    # calculate the length of the path
    length = 0
    for i in range(len(path) - 1):
        length += dis[path[i]][path[i + 1]]
    return length
```

5. Determine best path

```
# Step 2.1. Construct ant path solution
ant_path = []
ant_path_length = []
for i in range(pop):
    new_path = construct_path(dis, pheromone, alpha, beta)
    new_length = cal_dis(dis, new_path)
    ant_path.append(new_path)
    ant_path_length.append(new_length)
iter_best_path_length = min(ant_path_length)
if iter_best_path_length < best_length:
    best_length = iter_best_path_length
    best_path = ant_path[ant_path_length.index(iter_best_path_length)]
iter_best.append(best_length)
```

6. Update pheromones

```
# Step 2.2. Update pheromone
for i in range(stop_num):
    for j in range(i, stop_num):
        pheromone[i][j] *= (1 - pho)
        pheromone[j][i] *= (1 - pho)
for i in range(pop):
    delta = Q / ant_path_length[i]
    path = ant_path[i]
    for j in range(stop_num):
        pheromone[path[j]][path[j + 1]] += delta
        pheromone[path[j + 1]][path[j]] += delta
```

7. Plot best path, convergence graph, minimum distance

```
# Step 3. Sort the results
x = [i for i in range(iter)]
plt.figure()
plt.plot(x, iter_best, linewidth=2, color='blue')
plt.title("Convergence curve")
plt.xlabel("Iterations")
plt.ylabel('Global optimal value')
plt.show()

plt.figure()
plt.scatter(coord_x, coord_y, color='black')
for i in range(len(best_path) - 1):
    temp_x = [coord_x[best_path[i]], coord_x[best_path[i + 1]]]
    temp_y = [coord_y[best_path[i]], coord_y[best_path[i + 1]]]
    plt.plot(temp_x, temp_y, color='blue')
    plt.xlabel('latitude')
    plt.ylabel('longitude')
    plt.title('Best optimal route')
plt.show()
return {'Best path': best_path, 'Shortest length': best_length}
```

06

ALGORITHM PERFORMANCE

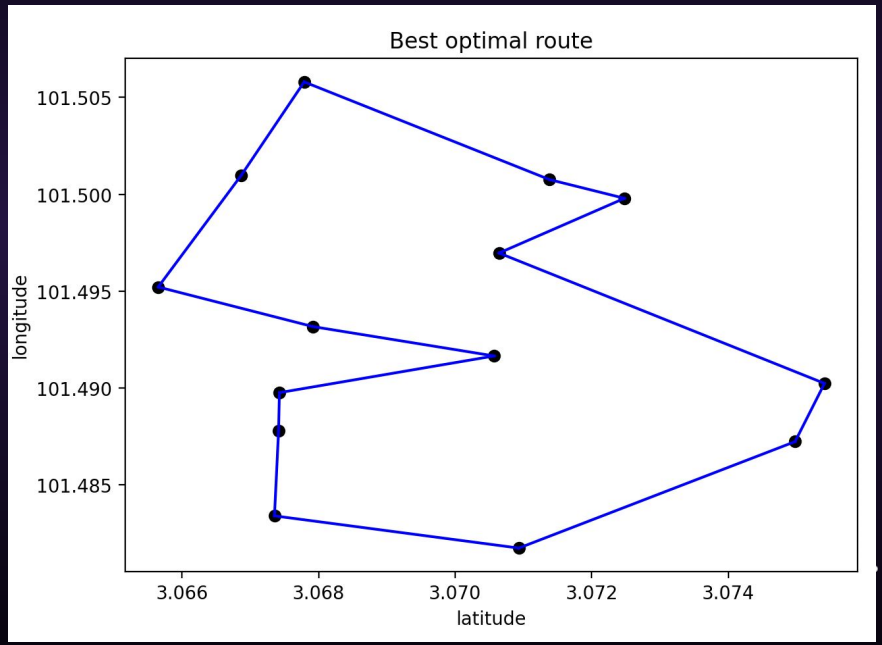
First Performance

Parameters

Iteration	5
Colony	100
Alpha	1
Beta	1
Rho	0.1

Shortest Distance: 0.0600421

Best Optimal Route



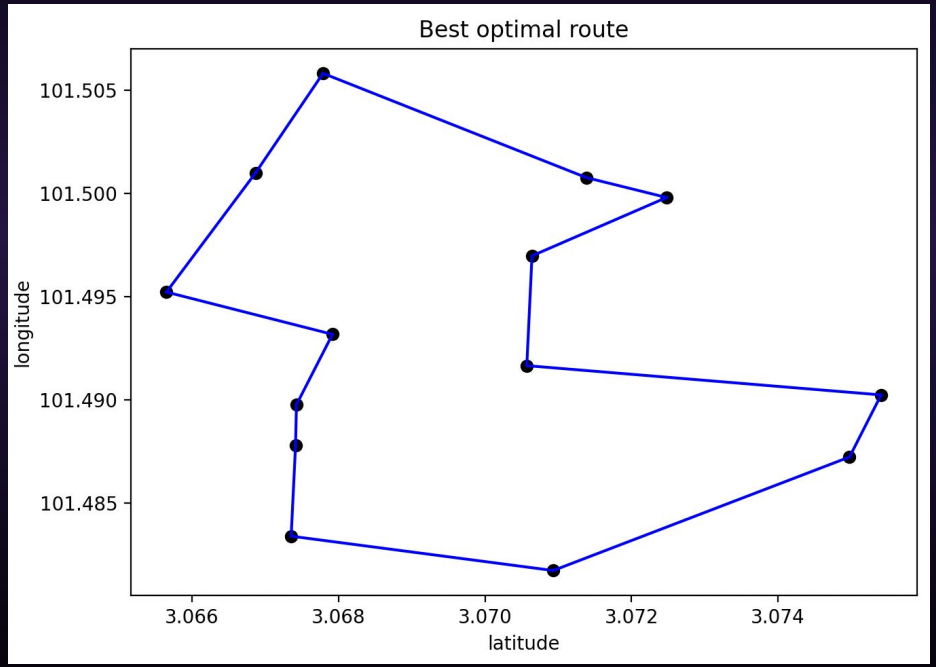
Second Performance

Parameters

Iteration	20
Colony	100
Alpha	1
Beta	1
Rho	0.1

Shortest Distance: 0.0588667

Best Optimal Route



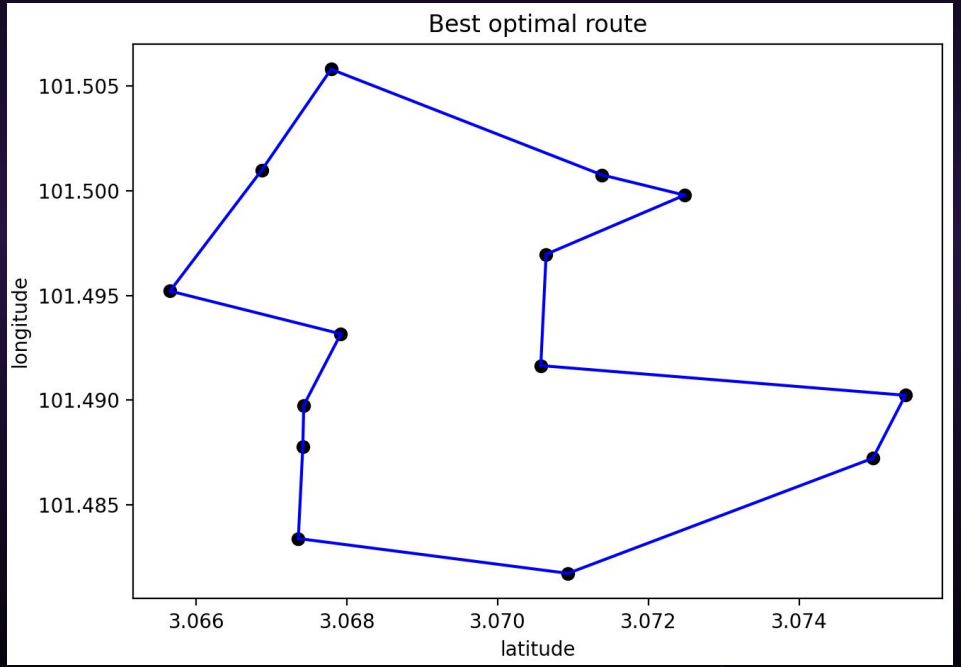
Third Performance

Parameters

Iteration	75
Colony	50
Alpha	1
Beta	1
Rho	0.5

Shortest Distance: 0.0588667

Best Optimal Route

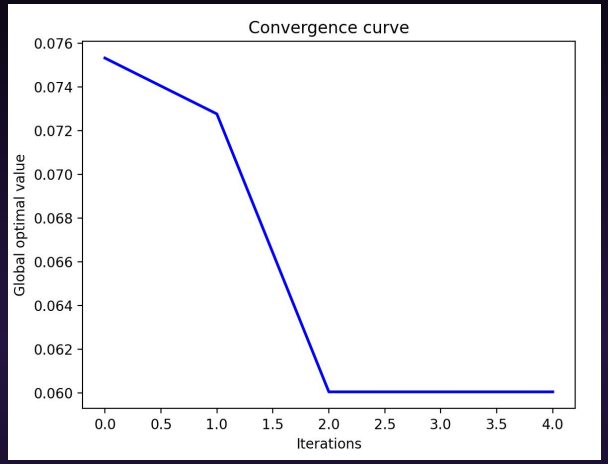


Overall Performance

Parameters/ Performance	First	Second	Third
Iteration	5	20	75
Colony	100	100	50
Alpha	1	1	1
Beta	1	1	1
Rho	0.1	0.1	0.5

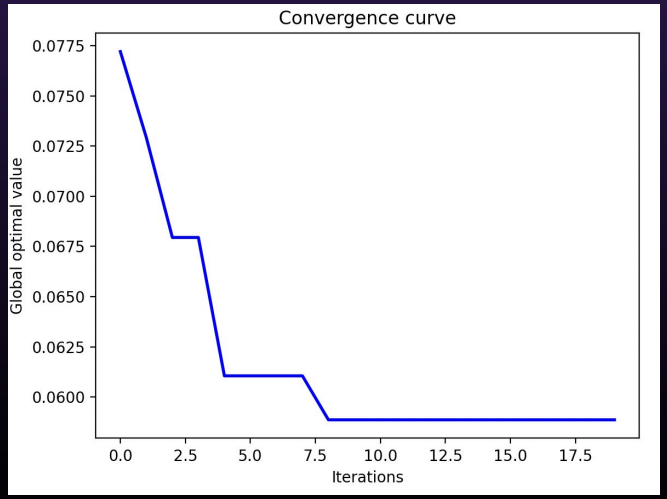
Minimum Distance	0.0600421	0.0588667	0.0588667

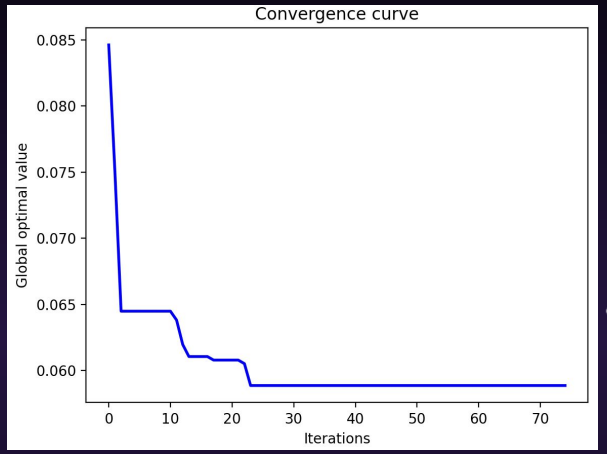
Performance Comparison



First performance

Second performance





Third performance

07

CONCLUSION

- Objective achieved, distance minimized
- Using latitude and longitude value of the bus stops
- Utilize Pythagoras Theorem and TSP to find the shortest distance between bus stops
- As the iterations increase, the minimum distance found decreases
- Before: 0.075 >> After: 0.059
- Different generations of parameter setting will produce different setting of final distance
- Potential implementation in UiTM Bus C's routing

THANK YOU