APPENDIX

CODING IMPORT

```
import pandas as pd
import matplotlib.pyplot as plt
from math import sin, cos, sqrt, atan2, radians
import numpy as np, random, operator
import folium
import seaborn as sns; sns.set()
from deap import base, creator, tools
import copy
import random
random.seed(3)
```

DISPLAYING ROUTE

```
hitung=0
for index, location in data.iterrows():
  if hitung!=119:
    loc = [(data["y"][index], data["x"][index]),
           (data["y"][index+1], data["x"][index+1])]
    folium.PolyLine(loc,
                    color='red',
                    weight=3,
                    opacity=1).add_to(m)
  else:
    break
  hitung=hitung+1
m
```

DISTANCE CALCULATION

```
count=0
pipelength=0
for index, location in data.iterrows():
  R = 6371.0
  if count != 119:
    lat1 = radians(data["x"][index])
    lon1 = radians(data["y"][index])
    lat2 = radians(data["x"][index+1])
    lon2 = radians(data["y"][index+1])
    count=count+1
  else:
    lat1 = radians(data["x"][index])
    lon1 = radians(data["y"][index])
    lat2 = radians(data["x"][index-119])
    lon2 = radians(data["y"][index-119])

  dlon = lon2 - lon1
  dlat = lat2 - lat1

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

```
    distance = R * c
    pipelength = pipelength + distance

    print("Result ",[index],":", distance)
    print("Pipelength: ",pipelength)
```

GENERATING INITIAL POPULATION

```
num_node=484

coordinates=[]
for index, location in data.iterrows():
  x=data["x"][index]
  y=data["y"][index]
  coordinates.append([x,y])
names = [i for i in range(num_homes)]
coordinates_dict = {name: coord for name,coord in zip(names, coordinates)}

coordinates=np.array(coordinates)

print(coordinates_dict)

def create_chromosome(_names):
    chromosome = copy.deepcopy(_names)
    np.random.shuffle(chromosome)
    return chromosome
```

CREATING AND EVALUATING CHROMOSOME

```
def evalChromosome(_distanceMatrix, _chromosome):
    distance = 0

    for i in range(len(_chromosome) - 1):
        _i = _chromosome[i]
        _j = _chromosome[i+1]
        distance += _distanceMatrix[_i][_j]

    distance += distanceMatrix[_chromosome[-1], _chromosome[0]]
    return distance,
```

PROCESSING GENETIC ALGORITHM

```
popul = tool.popul(n=populationNumber)
fitSet = list(tool.map(tool.eval, popul))
print(min(fitSet))
for ind, fit in zip(popul, fitSet):
    ind.fitness.values = fit

for gen in range(0, maxGeneration):

    if (gen % 10 == 0):
         print(f'Generation: {gen}' )
         print(f'Fitness: {fitness:}' )

```

```
    child = tool.select(popul, len(popul), tournsize=3)
    child = list(map(tool.clone, offspring))
for o1, o2 in zip(child[0::2], child[1::2]):
        if np.random.random() < crossoverprob:
            tool.crossover(o1, o2)
            del o1.fitness.values
            del o2.fitness.values
for chromosome in child:
    if np.random.random() < mutationprob:
       tool.mutate(chromosome, indpb=0.01)
       del chromosome.fitness.values
```

PAPER NAME

TA-19.K1.0007.docx

WORD COUNT	CHARACTER COUNT
7898 Words	**41400 Characters**
PAGE COUNT	FILE SIZE
20 Pages	**51.9KB**
SUBMISSION DATE	REPORT DATE
Dec 15, 2022 1:24 PM GMT+7	**Dec 15, 2022 1:24 PM GMT+7**

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 17% Internet database
- 11% Publications database
- Crossref database
- Crossref Posted Content database
- 5% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 10 words)