2021-2 Advanced Programming
Midterm Project

ELSEVIER

Safe distance-based vehicle routing
: Medical waste collection case study in COVID-19 pandemic

2016143509 Tong-Hun Lee (Yonsei Univ. Industrial Engineering)
2017144008 Beom-Seok Park (Yonsei Univ. Industrial Engineering)
2017147051 Jong-Woo Lee (Yonsei Univ Industrial Engineering)

Contents

- Overview
- Data Collection
- Data Pre-processing
- Optimization
- Results

Computers & Industrial Engineering 157 (2021) 107328

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

ELSEVIER

Safe distance-based vehicle routing problem: Medical waste collection case study in COVID-19 pandemic

Emre Eren [a,*], Umut Rıfat Tuzkaya [b]

[a] Department of Industrial Engineering, Faculty of Engineering and Architecture, University of Beykent, Turkey

[b] Department of Industrial Engineering, Faculty of Mechanical, University of Yildiz Teknik, Barbaros Avenue, PO Box: 34349, Besiktas, Istanbul, Turkey

ARTICLE INFO

Keywords:
Covid-19 and medical waste
Covid-19 pandemic
Medical waste
Membership function
Transportation of medical waste
Traveling salesman problem

ABSTRACT

In addition to the increasing population and rapid urbanization, the amount and variety of medical waste are rapidly increasing due to the coronavirus disease (COVID-19) pandemic affecting the whole world. COVID-19 does not only increase the amount of medical waste produced, medical wastes generated in the care of COVID-19 carries a high risk of transmission as well. In this regard, the safe and effective management of medical wastes has become a serious health and safety issue. This research aims to determine the safest and shortest transportation routes for medical waste vehicles. The safety scores used in this study were obtained in our previous study. The resulting safety scores were used in a multi-objective traveling salesman problem for deriving two objective functions, which are based on safety scores and total transportation distance. A conciliating solution was obtained by solving this linear programming model. The proposed model faced by health institutions in Istanbul has been applied for a specific district. According to the obtained results, suggestions for the direction of medical waste vehicles have been proposed.

1. Introduction

All types of wastes generated in healthcare institutions, research centers, and medical laboratories are referred to as healthcare wastes. 75–90% of wastes generated by healthcare service providers can be defined as domestic wastes, which are often regarded as "non-hazardous" or "general healthcare" wastes. In other words, wastes that do not pose any physical, chemical, biological or radioactive hazards. Such wastes are generated by the administrative, catering, cleaning, etc. services of healthcare institutions. The remaining portion of 10–25% is termed as "hazardous" healthcare wastes. World Health Organization (WHO) categorises this type of hazardous healthcare waste into seven main groups depending on their characteristics and risk levels, which are: pathological waste, infectious waste, sharps waste, radioactive waste, chemical waste, cytotoxic waste and pharmaceutical waste (Win et al., 2019). According to standard procedures in the medical field, the characteristics of healthcare wastes are similar in almost all countries. However, legal regulations regarding the safe management of medical waste may differ from one country to another. For example, according to USA regulations, used and unused implements, cultures and stocks of infectious agents, human blood and blood products, human pathological waste, and contaminated animal waste are referred to as medical waste (Mato & Kaseva, 1999). Another example, in China, medical waste is classified as chemical waste, medicine waste, injury waste, pathologic waste, and infectious waste (He, Li, & Fang, 2016). In Turkey, published in January 2017 by the Medical Waste Control Regulation, sharps waste, pathological waste and infectious waste are classified as medical waste. In this study, transport of medical waste in Turkey is discussed.

As a popular subject, Medical Waste Management (MWM) has been addressed with a variety of methods. Survey studies have been performed on detection, MWM generated by hospitals in various countries and cities. The amount of medical wastes generated has been reported to be 0.59 kg/(bed.day) for the European side, and 0,6199 kg/(bed.day) for the Asian side of Istanbul by Alagoz and Kocasoy (2008a); 0.63 kg/(bed.day) by Birpinar, Bilgili, and Erdogan (2009); 0.68 kg/(bed.day) by Yong, Gang, Guanxing, Tao, and Dawei (2009); and 0.89 kg/(bed.day) by Rolewicz-Kalińska (2016), and reportedly the amount of medical wastes increases each passing year. In these studies conducted before COVID-19 pandemic, it has been observed that the amount of medical waste generated in the light of factors such as urbanisation, industrialisation and population growth is increasing day by day. With today's COVID-19 outbreak, the number of patients in hospitals is increasing

Overview

Medical Waste Management(MWM)

MWM is an essential part of controlling an infectious epidemic like COVID-19.

Table 1
Hospital distances.

H (km)	H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	H_{12}	H_{13}	H_{14}	H_{15}
H_0	0	38,7	41,2	42,5	42,8	42,1	44,6	46,5	46,5	49,8	53,5	68,6	72,3	78	80,6	91
H_1	38,7	0	1,8	5,2	5,3	8,8	4,8	6,6	7	10,1	29,9	32,1	37	42,8	45,4	60,4
H_2	41,2	1,8	0	5,2	5,4	9,2	4,4	6,3	6,2	9,8	31	32,6	46,1	43,2	45,8	60,8
H_3	42,5	5,2	5,2	0	0,75	5,2	3,6	6,2	6,2	9,5	26,7	28,3	33,2	39	41,6	64,5
H_4	42,8	5,3	5,4	0,75	0	5,8	2,7	4,4	5,6	9,8	27	28,6	42,1	39,3	41,9	57,4
H_5	42,1	8,8	9,2	5,2	5,8	0	9,6	10,7	10,7	14,8	32	33,6	47,1	44,3	46,9	56
H_6	44,6	4,8	4,4	3,6	2,7	9,6	0	3	3,4	6,7	23,8	25,5	30,4	36,2	38,8	47,9
H_7	46,5	6,6	6,3	6,2	4,4	10,7	3	0	1,6	4,9	24	25,7	39,2	36,3	39	48,1
H_8	46,5	7	6,2	6,2	5,6	10,7	3,4	1,6	0	3,8	24,3	25,9	40,4	36,6	39,2	48,3
H_9	49,8	10,1	9,8	9,5	9,8	14,8	6,7	4,9	3,8	0	18,5	20,2	33,7	30,8	33,5	42,6
H_{10}	53,5	29,9	31	26,7	27	32	23,8	24	24,3	18,5	0	14,2	26,5	23,6	27,4	35,3
H_{11}	68,6	32,1	32,6	28,3	28,6	33,6	25,5	25,7	25,9	20,2	14,2	0	2,3	2,3	2	11,2
H_{12}	72,3	37	46,1	33,2	42,1	47,1	30,4	39,2	40,4	33,7	26,5	2,3	0	3,7	2,5	9
H_{13}	78	42,8	43,2	39	39,3	44,3	36,2	36,3	36,6	30,8	23,6	2,3	3,7	0	1,7	13,6
H_{14}	80,6	45,4	45,8	41,6	41,9	46,9	38,8	39	39,2	33,5	27,4	2	2,5	1,7	0	11
H_{15}	91	60,4	60,8	64,5	57,4	56	47,9	48,1	48,3	42,6	35,3	11,2	9	13,6	11	0

* H_1: Hospital 1, H_2: Hospital 2, H_3: Hospital 3, H_4: Hospital 4, H_5: Hospital 5, H_6: Hospital 6, H_7: Hospital 7, H_8: Hospital 8, H_9: Hospital 9, H_{10}: Hospital 10, H_{11}: Hospital 11, H_{12}: Hospital 12, H_{13}: Hospital 13, H_{14}: Hospital 14, H_{15}: Hospital 15.

<Hospital Distances>

Table 2
Hospital safety scores (Eren & Tuzkaya, 2019).

Hospital (H)	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	H_{12}	H_{13}	H_{14}	H_{15}
Hospital safety scores (S)	5,02	7,72	6,67	7,68	5,42	8,75	6,28	5,66	8,01	6	6,99	7,13	9,01	6,03	7,17

<Hospital Safety Scores>

$$\max \lambda$$

Limitations:

$$W_{1m} = \sum_{i}^{N} \sum_{j}^{N} d_{ij}.y_{ij}$$

$$W_{1g} = \sum_{i}^{N} \sum_{j}^{N} S_{ij}.x_{ij}$$

$$\boxed{\begin{array}{l} \dfrac{W_{1m}^{max} - W_{1m}}{W_{1m}^{max} - W_{1m}^{min}} \le \lambda \\[2ex] \dfrac{W_{1g} - W_{1g}^{min}}{W_{1g}^{max} - W_{1g}^{min}} \le \lambda \end{array}}$$

Fuzzy Optimization

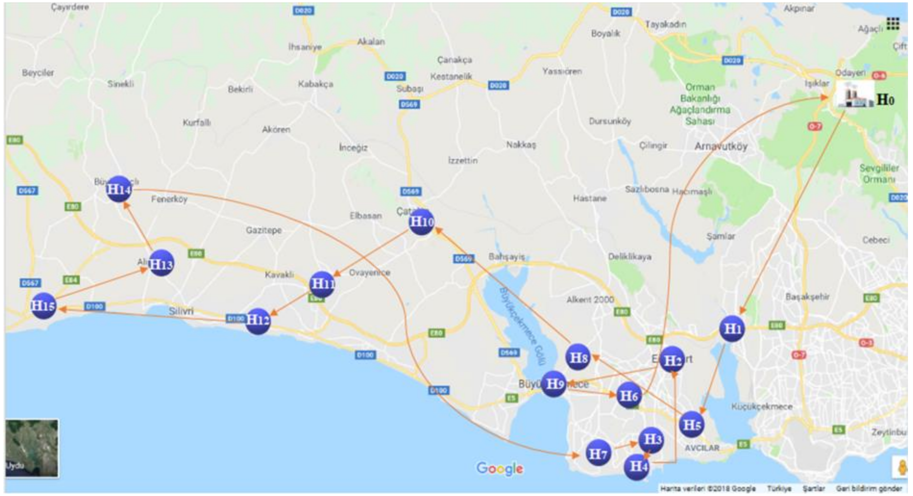
$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_j$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall_i$$

$$\sum_{j \in S} . \sum_{i \in S} x_{ij} \quad \le |S| - 1 \quad \forall S \subset N, \quad |S| \ge 2$$

$$y_{ij} \in \{0, 1\} \qquad \forall_{i,j}$$

<Optimization with membership functions>



<The Optimum tour of MWC Vehicle>

Data Collection

Our group decided to apply the methodology used in this study to Korean hospitals. Since the data presented in this paper is for Turkish hospitals, we reconstructed it into Korean hospitals data.

Hospital Distances

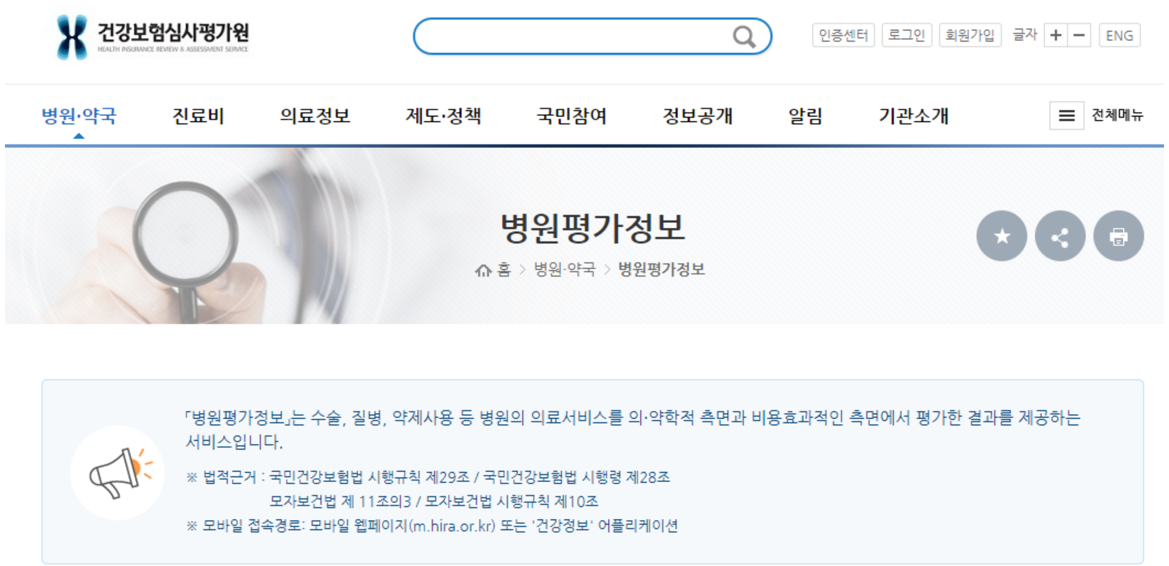
Extracting latitude and longitude of hospital with Google Map api based on hospital address.

고려대학교의과대학부속병원(안암병원)
가톨릭대학교여의도성모병원
가톨릭대학교은평성모병원
강동경희대학교의대병원
강북삼성병원
건국대학교병원
경희대학교병원
고려대학교의과대학부속구로병원
국립중앙의료원
노원을지대학교병원
삼성서울병원
삼육서울병원
서울대학교병원
서울특별시보라매병원
서울특별시서울의료원
성심의료재단강동성심병원
성애의료재단성애병원
순천향대학교부속서울병원
에이치플러스양지병원
연세대학교의과대학강남세브란스병원
의료법인한전의료재단한일병원
이화여자대학교의과대학부속목동병원
이화여자대학교의과대학부속서울병원
인제대학교상계백병원
재단법인아산사회복지재단서울아산병원
중앙대학교병원
학교법인가톨릭학원가톨릭대학교서울성모병원
학교법인연세대학교의과대학세브란스병원
한국보훈복지의료공단중앙보훈병원
한림대학교 강남성심병원
한양대학교병원

<List of advanced general hospitals(상급종합병원) in Seoul>

Safety Scores

Using the safety score evaluated by HEALTH INSURANCE REVIEW & ASSESSMENT SERVICE



<HEALTH INSURANCE REVIEW & ASSESSMENT SERVICE>

Data Collection

Our group decided to apply the methodology used in this study to Korean hospitals. Since the data presented in this paper is for Turkish hospitals, we reconstructed it into Korean hospitals data.

hospital_data.csv

	주소	병원분류	병원경도	병원위도	score
0	서울특별시 성북구 고려대로 73 고려대병원 (안암동5가)	A	127.026471	37.587156	84
1	서울특별시 영등포구 63로 10 여의도성모병원 (여의도동)	A	126.936731	37.518272	84
2	서울특별시 은평구 통일로 1021 (진관동)	A	126.916151	37.633608	89
3	서울특별시 강동구 동남로 892 (상일동)	A	127.157522	37.553476	89
4	서울특별시 종로구 새문안로 29 (평동)	A	126.967938	37.568498	85
5	서울특별시 광진구 능동로 120-1 (화양동)	A	127.072123	37.540845	84
6	서울특별시 동대문구 경희대로 23 (회기동)	A	127.051832	37.593877	83
7	서울특별시 구로구 구로동로 148 고려대부속구로병원 (구로동)	A	126.884745	37.492111	80
8	서울특별시 중구 을지로 245 (을지로6가)	A	127.005795	37.567340	85
9	서울특별시 노원구 한글비석로 68 을지병원 (하계동)	A	127.070003	37.636443	81
10	서울특별시 강남구 일원로 81 (일원동 삼성의료원)	A	127.086682	37.488516	88
11	서울특별시 동대문구 망우로 82 (휘경동)	A	127.065329	37.587992	80
12	서울특별시 종로구 대학로 101 (연건동)	A	126.998963	37.579666	83
13	서울특별시 동작구 보라매로5길 20 (신대방동)	A	126.924049	37.493718	82
14	서울특별시 중랑구 신내로 156 (신내동)	A	127.098091	37.612869	81

⋮

Data Pre-processing

Import

```python
import math
import numpy as np
import gurobipy as gp
from gurobipy import GRB
from gurobipy import quicksum
import veroviz as vrv


# API Key for ORS geographical data to provide road network
# Website Link: https://openrouteservice.org/dev/#/home
ORS_API_KEY = '5b3ce3597851110001cf6248048e439824f5449991dafb54db9718c7'
```

Data Pre-processing

Load Hospital Data

```python
# Hospital data loading
# Data preprocessing after loading is necessary
H = np.genfromtxt('C:\Advanced Programming\hospital_data.csv', dtype=None, delimiter=",", encoding='UTF-8')
coord_data = np.transpose(H)

# Making a list of integrated coordinates of every tertiary general hospital
hospital_coord_str = []
for num in range(1, coord_data.shape[1]):
    if coord_data[4][num] in ['G001', 'G006', 'G099']:
        hospital_coord_str.append([coord_data[-3][num], coord_data[-4][num]])

# Converting elements of hospital coordinates list from string to float data type
hospital_coord = []
for str_list in hospital_coord_str:
    float_list = list(map(float, str_list))
    hospital_coord.append(float_list)
```

Data Pre-processing

Load Safety Score

```python
# Converting elements of hospital safety scores list from string to float data type
hospital_safety = []
for string in hospital_safety_str:
    hospital_safety.append(int(string))

# Specifying list of hospitals
hospital_list = []
for i in range(len(hospital_coord)):
    hospital_list.append('H' + str(i+1))
```

Data Pre-processing

Create Hospital Nodes

```python
# Create hospital nodes
myNodes = vrv.createNodesFromLocs(locs=hospital_coord, leafletIconPrefix='fa', leafletIconType='ambulance')

# Create time matrix and distance matrix in a dictionary form
[timeSec, distMeters] = vrv.getTimeDist2D(nodes        = myNodes,
                                          outputDistUnits  = 'km',
                                          routeType    = 'fastest',
                                          dataProvider = 'ORS-online',
                                          dataProviderArgs = { 'APIkey' : ORS_API_KEY })
```

Data Pre-processing

The matrix of distance between hospitals and Safety Score Matrix

```python
# Converting distance matrix between hospitals into 2D array structure
dist_data = np.array(list(distMeters.values()))
d_size = int(math.sqrt(len(distMeters)))
d_shape = (d_size, d_size)
dist_matrix = dist_data.reshape(d_shape)

# Create safety score matrix
safety_matrix = np.zeros((len(hospital_safety), len(hospital_safety)))
for i in range(len(hospital_safety)):
    for j in range(len(hospital_safety)):
        if i == j:
            safety_matrix[i][j] = 0
        else:
            safety_matrix[i][j] = hospital_safety[i] * hospital_safety[j] * 0.01
```

Optimization

Modelling

```
# Constructing data structure for optimization model
n = len(hospital_list)
hospitals = range(n)
hospital = range(1, n)
dist_dict = {(i, j): dist_matrix[i][j] for i in hospitals for j in hospitals}
safety_dict = {(i, j): safety_matrix[i][j] for i in hospitals for j in hospitals}
```

```
# Set model(distance)
md = gp.Model('Waste_VRP_distance')

# Decision Variables(distance)
y_vars = md.addVars(dist_dict.keys(), obj=dist_dict, vtype=GRB.BINARY, name='y')
u_vars = md.addVars(n)
```

Optimization

Application of Safety Scores in the Travelling Salesman Problem

```
# Constraints(distance)
md.addConstrs(quicksum(y_vars[i, j] for j in hospitals if i != j) == 1 for i in hospitals)
md.addConstrs(quicksum(y_vars[i, j] for i in hospitals if i != j) == 1 for j in hospitals)
md.addConstrs(u_vars[i] - u_vars[j] + n*y_vars[i, j] <= n-1 for i in hospital for j in hospital)
md.addConstrs(u_vars[i] <= n-1 for i in hospital)
md.addConstrs(u_vars[i] >= 0 for i in hospital)

# The objective is to minimize the total distance
md.modelSense = GRB.MINIMIZE

# Optimize model(distance)
md.optimize()
```

$$Z_{1min} = \sum_{i}^{N} \sum_{j}^{N} d_{ij} . y_{ij} \qquad (12)$$

Limitations:

$$\sum_{i=1}^{N} y_{ij} = 1 \qquad \forall_j \qquad (13)$$

$$\sum_{j=1}^{N} y_{ij} = 1 \qquad \forall_i \qquad (14)$$

$$\sum_{j \in S} . \sum_{i \in S} y_{ij} \leq |S| - 1 \qquad \forall S \subset N, \qquad |S| \geq 2 \qquad (15)$$

$$y_{ij} \in \{0, 1\} \qquad \forall_{i,j} \qquad (16)$$

Optimization

Application of Safety Scores in the Travelling Salesman Problem

```python
# Set model(safety)
ms = gp.Model('Waste_VRP_safety')

# Decision Variables(safety)
x_vars = ms.addVars(safety_dict.keys(), obj=safety_dict, vtype=GRB.BINARY, name='x')
u_vars = ms.addVars(n)

# Constraints(safety)
ms.addConstrs(quicksum(x_vars[i, j] for j in hospitals if i != j) == 1 for i in hospitals)
ms.addConstrs(quicksum(x_vars[i, j] for i in hospitals if i != j) == 1 for j in hospitals)
ms.addConstrs(u_vars[i] - u_vars[j] + n*x_vars[i, j] <= n-1 for i in hospital for j in hospital)
ms.addConstrs(u_vars[i] <= n-1 for i in hospital)
ms.addConstrs(u_vars[i] >= 0 for i in hospital)

# The objective is to maximize the safety scores
ms.modelSense = GRB.MAXIMIZE

# Optimize model(safety)
ms.optimize()
```

$$Z_{2max} = \sum_{i}^{N} \sum_{j}^{N} S_{ij}.x_{ij} \tag{17}$$

Limitations:

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_j \tag{18}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall_i \tag{19}$$

$$\sum_{j\in S}.\sum_{i\in S} x_{ij} \quad \leq |S| - 1 \quad \forall S \subset N, \quad |S| \geq 2 \tag{20}$$

$$x_{ij} \in \{0, 1\} \qquad \forall_{i,j} \tag{21}$$

Optimization

Optimization with Membership Function

```python
# Set model(fuzzy)
mf = gp.Model('Waste_VRP_fuzzy')

# Decision Variables(fuzzy)
x_vars = mf.addVars(safety_dict.keys(), vtype=GRB.BINARY, name='x')
u_vars = mf.addVars(n)
w1m_var = mf.addVar()
w1g_var = mf.addVar()
lambda_var = mf.addVar(obj=1, name='lambda')
```

$$\max \lambda \tag{24}$$

Limitations:

$$W_{1m} = \sum_{i}^{N} \sum_{j}^{N} d_{ij} . y_{ij} \tag{25}$$

$$W_{1g} = \sum_{i}^{N} \sum_{j}^{N} S_{ij} . x_{ij} \tag{26}$$

$$\frac{W_{1m}{}^{max} - W_{1m}}{W_{1m}{}^{max} - W_{1m}{}^{min}} \leq \lambda \tag{27}$$

$$\frac{W_{1g} - W_{1g}{}^{min}}{W_{1g}{}^{max} - W_{1g}{}^{min}} \leq \lambda \tag{28}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_j \tag{29}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall_i \tag{30}$$

$$\sum_{j \in S} . \sum_{i \in S} x_{ij} \quad \leq |S| - 1 \quad \forall S \subset N, \quad |S| \geq 2 \tag{31}$$

$$y_{ij} \in \{0, 1\} \qquad \forall_{i,j} \tag{32}$$

Optimization

Optimization with Membership Function

```
# Constraints(fuzzy)
mf.addConstr(w1m_var == quicksum(dist_dict[i, j] * x_vars[i, j] for j in hospitals for i in hospitals if i != j))
mf.addConstr(w1g_var == quicksum(safety_dict[i, j] * x_vars[i, j] for j in hospitals for i in hospitals if i != j))

mf.addConstr(lambda_var <= (d_max - w1m_var)/(d_max - d_min))
mf.addConstr(lambda_var <= (w1g_var - s_min)/(s_max - s_min))

mf.addConstrs(quicksum(x_vars[i, j] for j in hospitals if i != j) == 1 for i in hospitals)
mf.addConstrs(quicksum(x_vars[i, j] for i in hospitals if i != j) == 1 for j in hospitals)
mf.addConstrs(u_vars[i] - u_vars[j] + n*x_vars[i, j] <= n-1 for i in hospital for j in hospital)
mf.addConstrs(u_vars[i] <= n-1 for i in hospital)
mf.addConstrs(u_vars[i] >= 0 for i in hospital)

# The objective is to maximize the general satisfaction level
mf.modelSense = GRB.MAXIMIZE

# Optimize model(fuzzy)
mf.optimize()
```

$$\max \lambda \tag{24}$$

Limitations:

$$W_{1m} = \sum_{i}^{N} \sum_{j}^{N} d_{ij} . y_{ij} \tag{25}$$

$$W_{1g} = \sum_{i}^{N} \sum_{j}^{N} S_{ij} . x_{ij} \tag{26}$$

$$\frac{W_{1m}{}^{max} - W_{1m}}{W_{1m}{}^{max} - W_{1m}{}^{min}} \leq \lambda \tag{27}$$

$$\frac{W_{1g} - W_{1g}{}^{min}}{W_{1g}{}^{max} - W_{1g}{}^{min}} \leq \lambda \tag{28}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_{j} \tag{29}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall_{i} \tag{30}$$

$$\sum_{j \in S} . \sum_{i \in S} x_{ij} \quad \leq |S| - 1 \quad \forall S \subset N, \quad |S| \geq 2 \tag{31}$$

$$y_{ij} \in \{0, 1\} \qquad \forall_{i,j} \tag{32}$$

Results

Making Routes for each Optimization

```python
# Making routes for each optimization problem (Distance, Safety score, Multi-objective)
vehicle_routes = [[0], [0], [0]]
decision_lists = [[], [], []]

for i in range(len(vehicle_routes)):
    if i == 0:
        for v in md.getVars():
            if round(v.x) == 1 and v.varName.startswith('y'):
                decision_lists[i].append(eval(v.varName[1:]))
    elif i == 1:
        for v in ms.getVars():
            if round(v.x) == 1 and v.varName.startswith('x'):
                decision_lists[i].append(eval(v.varName[1:]))
    elif i == 2:
        for v in mf.getVars():
            if round(v.x) == 1 and v.varName.startswith('x'):
                decision_lists[i].append(eval(v.varName[1:]))

for i in range(len(vehicle_routes)):
    for num in range(len(decision_lists[i])):
        for num in range(len(decision_lists[i])):
            if decision_lists[i][num][0] in vehicle_routes[i] and decision_lists[i][num][1] not in vehicle_routes[i]:
                vehicle_routes[i].append(decision_lists[i][num][1])
    vehicle_routes[i].append(0)
```

Results

Redifining Solution Route

```python
# Redefined solution route to match the data structure of Veroviz module
solution_routes = [[], [], []]
for i in range(len(solution_routes)):
    for j in range(len(vehicle_routes[i]) - 1):
        solution_routes[i].append([vehicle_routes[i][j]+1, vehicle_routes[i][j+1]+1])
```

Results

Visualization

```python
# Visualization of the final solution route
for i in range(len(vehicle_routes)):
    mySolution = {
        'VRP': solution_routes[i]
    }

    myAssignments = vrv.initDataframe('assignments')

    if i == 0:
        vehicleProperties = {
                'VRP': {'model': 'veroviz/models/car_red.gltf',
                        'leafletColor': 'red'}
        }
    elif i == 1:
        vehicleProperties = {
            'VRP': {'model': 'veroviz/models/car_red.gltf',
                        'leafletColor': 'blue'}
        }
    elif i == 2:
        vehicleProperties = {
            'VRP': {'model': 'veroviz/models/car_red.gltf',
                        'leafletColor': 'green'}
        }
```

```python
    for v in mySolution:
        endTimeSec = 0.0
        for arc in mySolution[v]:
            [myAssignments, endTimeSec] = vrv.addAssignment2D(
                initAssignments=myAssignments,
                objectID=v,
                modelFile=vehicleProperties[v]['model'],
                startLoc=list(myNodes[myNodes['id'] == arc[0]][['lat', 'lon']].values[0]),
                endLoc=list(myNodes[myNodes['id'] == arc[1]][['lat', 'lon']].values[0]),
                startTimeSec=endTimeSec,
                leafletColor=vehicleProperties[v]['leafletColor'],
                routeType='fastest',
                dataProvider='ORS-online',
                dataProviderArgs={'APIkey': ORS_API_KEY})

    if i == 0:
        myMap = vrv.createLeaflet(nodes=myNodes, arcs=myAssignments, mapFilename="seoul_vrp_distance.html")
    elif i == 1:
        myMap = vrv.createLeaflet(nodes=myNodes, arcs=myAssignments, mapFilename="seoul_vrp_safety.html")
    elif i == 2:
        myMap = vrv.createLeaflet(nodes=myNodes, arcs=myAssignments, mapFilename="seoul_vrp_fuzzy.html")
```

Result

Total Distance, Total Safety Score

```python
# Calculating total distance of the tour
def total_dist(x):
    distance = np.zeros(3)
    for j in range(len(vehicle_routes[x]) - 1):
        distance[x] += dist_matrix[vehicle_routes[x][j], vehicle_routes[x][j+1]]
    return round(distance[x], 3)


# Calculating total safety score of the tour
def total_score(x):
    safety_score = np.zeros(3)
    for j in range(len(vehicle_routes[x]) - 1):
        safety_score[x] += safety_matrix[vehicle_routes[x][j], vehicle_routes[x][j+1]]
    return round(safety_score[x], 2)
```

Result

Total Distance, Total Safety Score

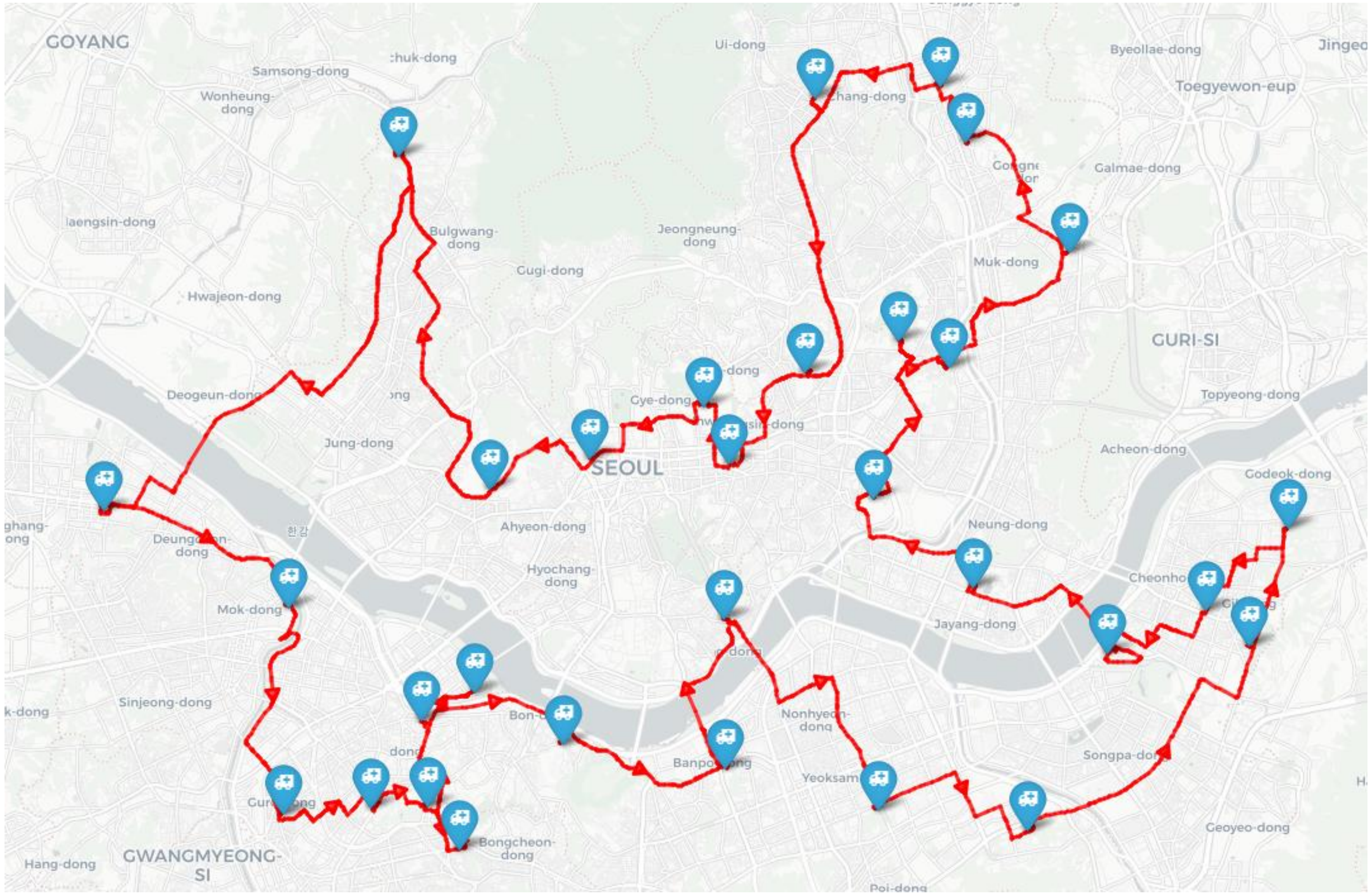
```
-----------------------------------------------------------------------------------------
< Minimize total distance >
Optimal route:  [0, 8, 12, 4, 27, 2, 22, 21, 7, 29, 13, 18, 16, 1, 25, 26, 17, 19, 10, 28, 3, 15, 24, 5, 30, 6, 11, 14, 9, 23, 20, 0]
Total distance:  155.373 km
Total safety score:  2176.43 points

< Maximize total safety score >
Optimal route:  [0, 7, 21, 16, 11, 28, 14, 23, 18, 9, 20, 13, 12, 17, 19, 4, 22, 8, 30, 27, 26, 3, 24, 2, 25, 10, 1, 5, 15, 6, 29, 0]
Total distance:  476.308 km
Total safety score:  2178.14 points

< Maximize general satisfaction level >
Optimal route:  [0, 6, 20, 23, 9, 14, 11, 28, 15, 3, 24, 10, 26, 25, 27, 2, 22, 21, 16, 7, 18, 13, 29, 1, 4, 8, 12, 17, 19, 5, 30, 0]
Optimal lambda value: 0.941013
Total distance:  188.486 km
Total safety score:  2177.85 points

Process finished with exit code 0
```

Structure Favorites

Run TODO Problems Terminal Python Packages Python Console

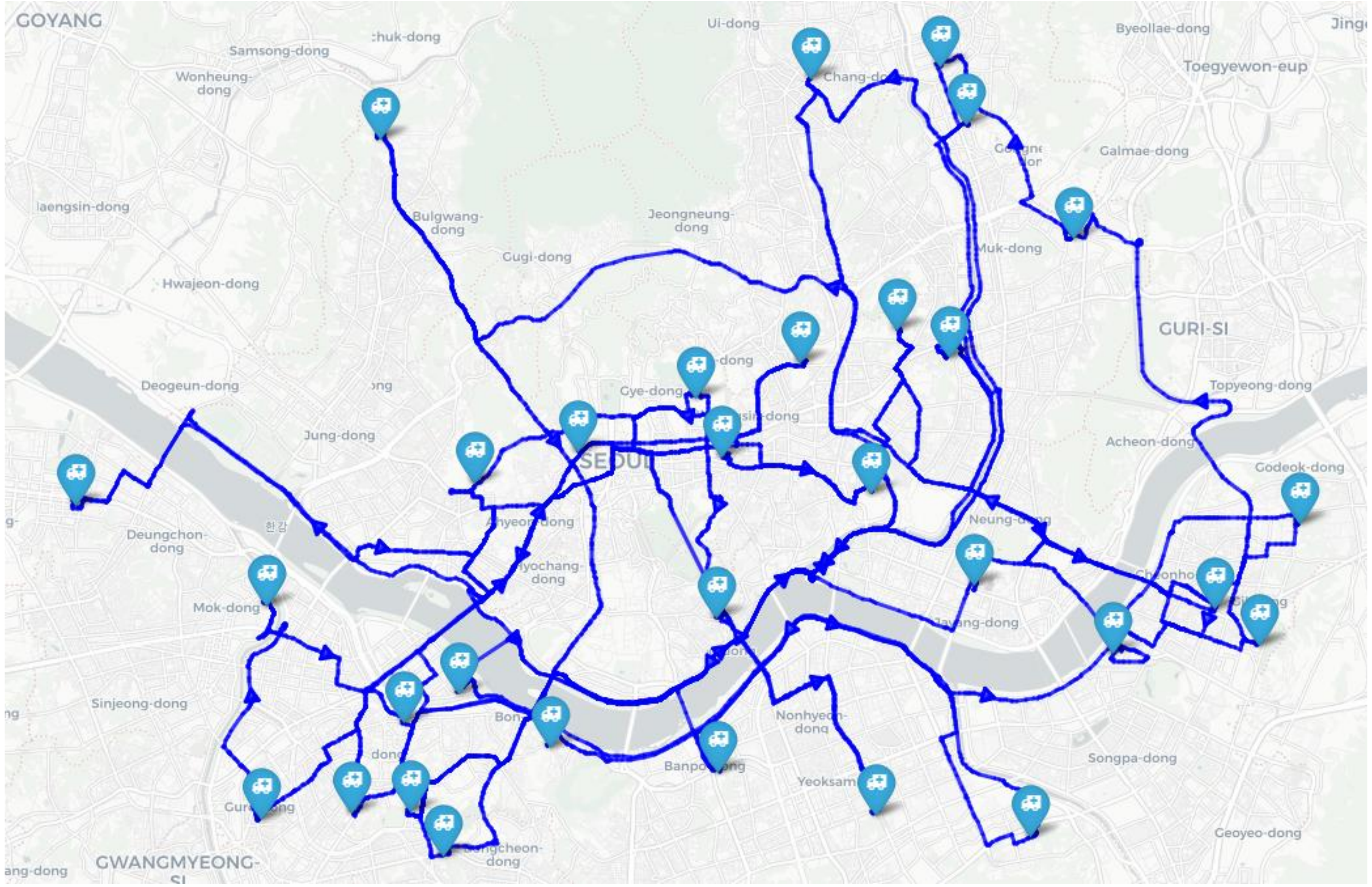
Results

seoul_vrp_distance.html



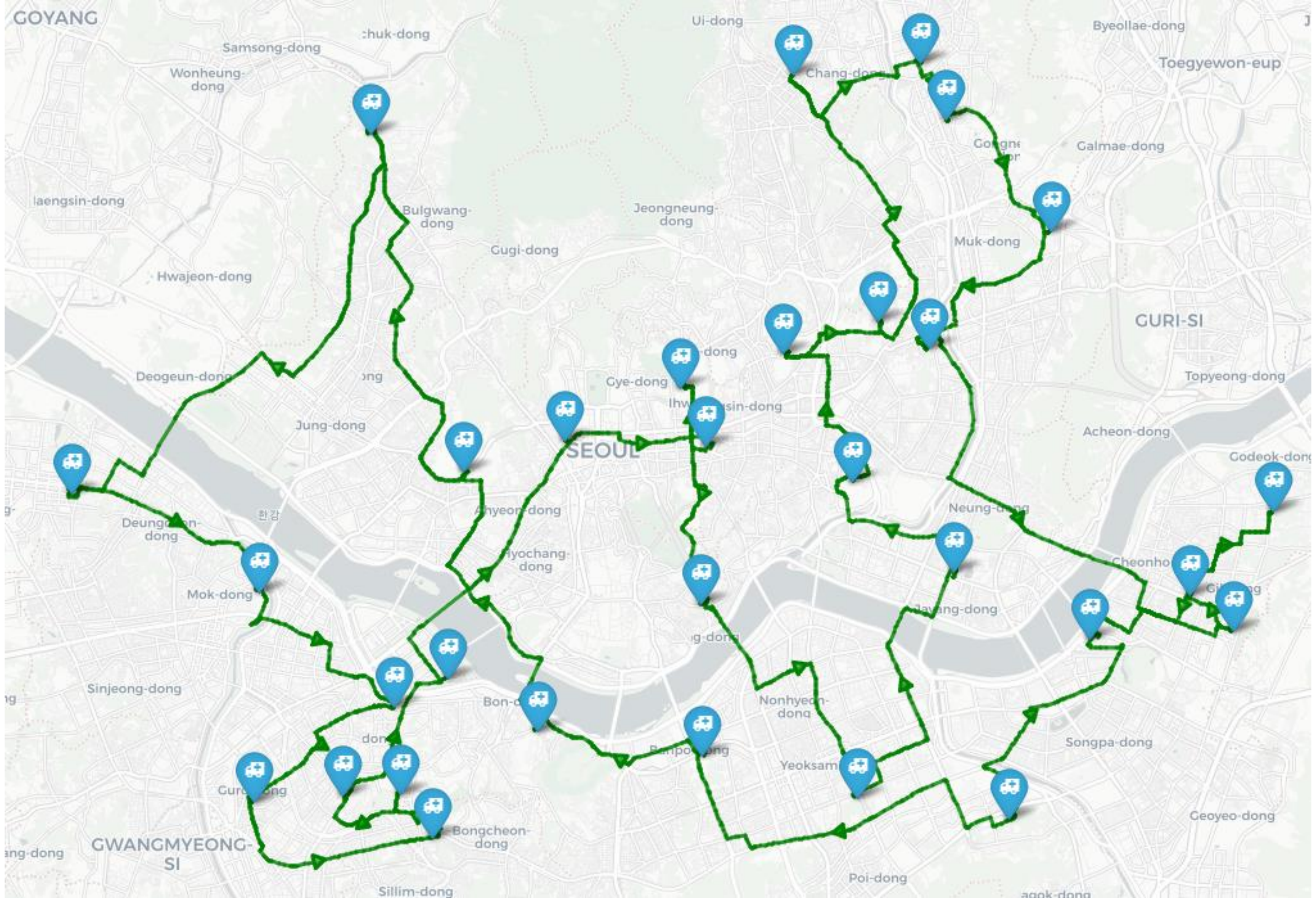
Results

seoul_vrp_safety.html



Results

seoul_vrp_fuzzy.html



Thank You!