HTS_barren_soils_case_of_latvia

July 16, 2024

[1]:
```
# Student: ALEKSEJS VESJOLIJS
# ST43537

# Date: 20 May 2024

# RELSTAT 2024
# Research: Hyperloop Routes Optimization Considering Barren Soil Using␣
↪Operation Research, Case of Latvia

# TRANSPORTA UN SAKARU INSTITŪTS
# DATORZINĀTŅU UN TELEKOMUNIKĀCIJU FAKULTĀTE
```

[2]:
```
# EXPLORATIVE DATA ANALYSIS
```

[3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pulp import LpMaximize, LpProblem, LpVariable, lpSum, LpStatus, value
```

[4]:
```
# Collected data from Copernicus
data = {
    'From': ['Riga', 'Riga', 'Riga', 'Riga', 'Daugavpils', 'Daugavpils',␣
↪'Daugavpils', 'Liepaja', 'Liepaja', 'Liepaja', 'Jelgava', 'Jelgava',␣
↪'Jelgava', 'Jurmala', 'Jurmala', 'Valmiera', 'Valmiera', 'Jekabpils',␣
↪'Jelgava', 'Rezekne', 'Jekabpils', 'Liepaja', 'Jurmala', 'Ventspils',␣
↪'Jelgava', 'Ventspils', 'Valmiera', 'Rezekne', 'Jekabpils'],
    'To': ['Jelgava', 'Jurmala', 'Valmiera', 'Jekabpils', 'Jelgava', 'Rezekne',␣
↪'Jekabpils', 'Jelgava', 'Jurmala', 'Ventspils', 'Jurmala', 'Ventspils',␣
↪'Valmiera', 'Riga', 'Valmiera', 'Riga', 'Jekabpils', 'Riga', 'Daugavpils',␣
↪'Daugavpils', 'Daugavpils', 'Liepaja', 'Liepaja', 'Liepaja', 'Jelgava',␣
↪'Jurmala', 'Rezekne', 'Valmiera', 'Valmiera'],
    'Barren soil': [13, 6, 27, 29, 37, 13, 9, 45, 55, 26, 15, 40, 46, 6, 27,␣
↪29, 4, 13, 37, 13, 9, 25, 55, 26, 15, 20, 46, 24, 4],
    'Distance': [45, 38, 107, 140, 230, 89, 89, 182, 192, 118, 45, 176, 185,␣
↪38, 107, 140, 162, 45, 230, 89, 89, 182, 192, 118, 45, 154, 185, 101, 162]
}
```

```python
df = pd.DataFrame(data)
```

[5]:
```python
print(df.head())
```

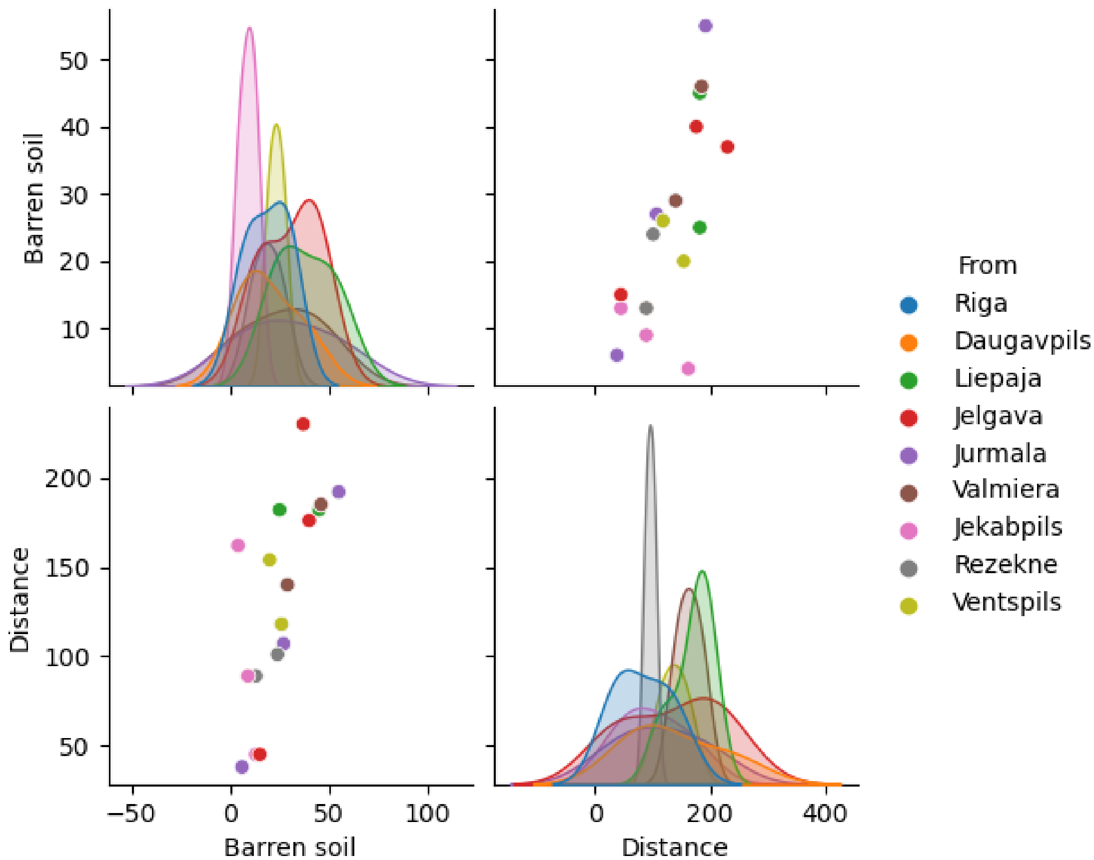
```
         From         To  Barren soil  Distance
0        Riga    Jelgava           13        45
1        Riga    Jurmala            6        38
2        Riga   Valmiera           27       107
3        Riga  Jekabpils           29       140
4  Daugavpils    Jelgava           37       230
```

[6]:
```python
print(df.describe())
```

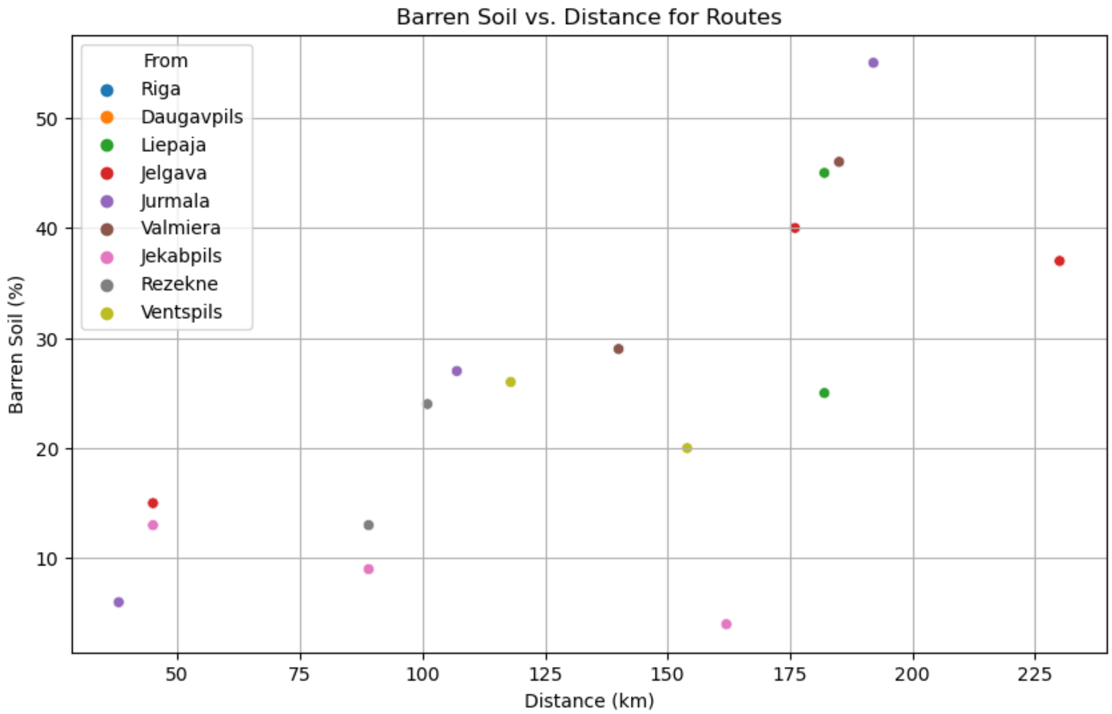
```
       Barren soil    Distance
count     29.00000   29.000000
mean      24.62069  126.724138
std       15.29569   59.315679
min        4.00000   38.000000
25%       13.00000   89.000000
50%       25.00000  118.000000
75%       37.00000  182.000000
max       55.00000  230.000000
```

[7]:
```python
# Create plots to visualize the data
sns.pairplot(df, hue='From')
plt.show()
```

```
d:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
d:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



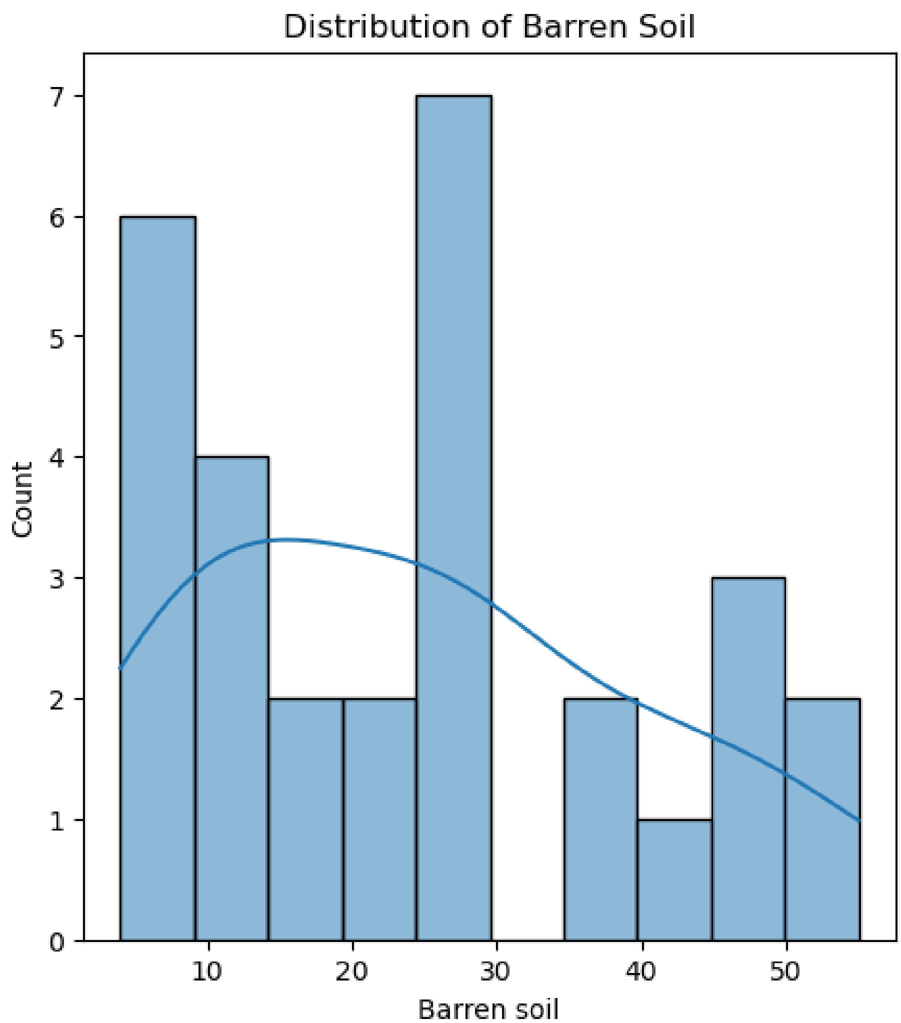
[8]:
```
# Plotting Barren soil vs. Distance
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Distance', y='Barren soil', hue='From', data=df)
plt.title('Barren Soil vs. Distance for Routes')
plt.xlabel('Distance (km)')
plt.ylabel('Barren Soil (%)')
plt.grid(True)
plt.show()
```



[9]:
```python
# Checking distribution of Barren Soil and Distance
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['Barren soil'], bins=10, kde=True)
plt.title('Distribution of Barren Soil')
```

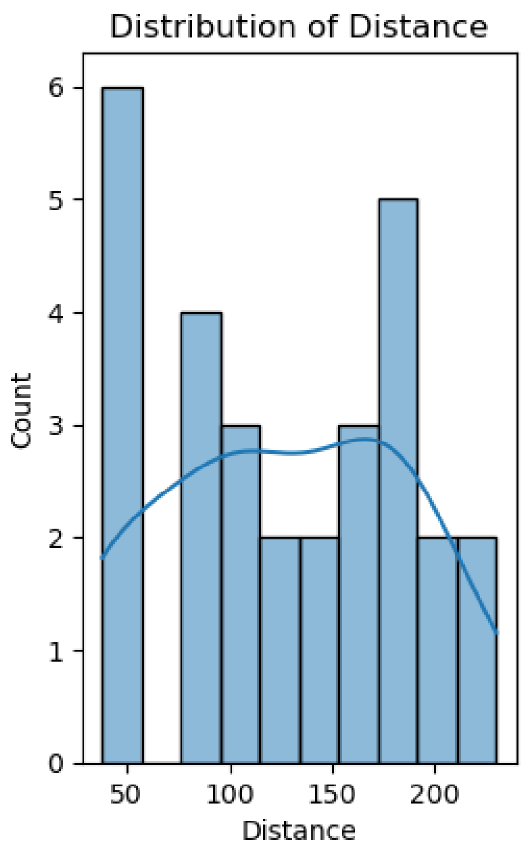
```
d:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[9]:
```
Text(0.5, 1.0, 'Distribution of Barren Soil')
```



[10]:
```
plt.subplot(1, 2, 2)
sns.histplot(df['Distance'], bins=10, kde=True)
plt.title('Distribution of Distance')
plt.show()
```

```
d:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[11]:
```
# OPERATIONS RESEARCH PROBLEM SOLUTION
```

[12]:
```
data = {
    'From': ['Riga', 'Riga', 'Riga', 'Daugavpils', 'Daugavpils', 'Liepaja',␣
↪'Jelgava', 'Jelgava', 'Valmiera', 'Jekabpils'],
    'To': ['Jelgava', 'Jurmala', 'Valmiera', 'Jelgava', 'Rezekne', 'Jelgava',␣
↪'Jurmala', 'Ventspils', 'Rezekne', 'Valmiera'],
    'Barren soil': [13, 6, 27, 37, 13, 45, 15, 40, 46, 24],
    'Distance': [45, 38, 107, 230, 89, 182, 45, 176, 185, 101]
}
df = pd.DataFrame(data)

# Create the linear programming model
model = LpProblem(name="hyperloop-route-optimization", sense=LpMaximize)

# Decision variables
route_vars = {row.Index: LpVariable(name=f"route_{row.Index}", cat='Binary')
              for row in df.itertuples()}
```

```python
# Objective function: Maximize the sum of barren soil * decision variable
model += lpSum(row['Barren soil'] * route_vars[idx] for idx, row in df.
 ↪iterrows()), "Maximize_Barren_Soil_Coverage"

# Constraints: each city is connected at least once
cities = set(df['From']) | set(df['To'])
for city in cities:
    model += (lpSum(route_vars[i] for i in df.index[(df['From'] == city) |␣
 ↪(df['To'] == city)]) >= 1,
              f"Connectivity_{city}")

# Solving the model
model.solve()

print("Status:", LpStatus[model.status])
print("Optimal Solution to the problem: Total Barren Soil = ", value(model.
 ↪objective))
for var in model.variables():
    if var.value() == 1:
        print(var.name, "=", var.value())

selected_routes = df.loc[[int(v.name.split('_')[1]) for v in model.variables()␣
 ↪if v.varValue == 1]]
print(selected_routes)
```

```
Status: Optimal
Optimal Solution to the problem: Total Barren Soil =  266.0
route_0 = 1.0
route_1 = 1.0
route_2 = 1.0
route_3 = 1.0
route_4 = 1.0
route_5 = 1.0
route_6 = 1.0
route_7 = 1.0
route_8 = 1.0
route_9 = 1.0
         From         To  Barren soil  Distance
0        Riga    Jelgava           13        45
1        Riga    Jurmala            6        38
2        Riga   Valmiera           27       107
3   Daugavpils    Jelgava           37       230
4   Daugavpils    Rezekne           13        89
5      Liepaja    Jelgava           45       182
6      Jelgava    Jurmala           15        45
7      Jelgava  Ventspils           40       176
8     Valmiera    Rezekne           46       185
```

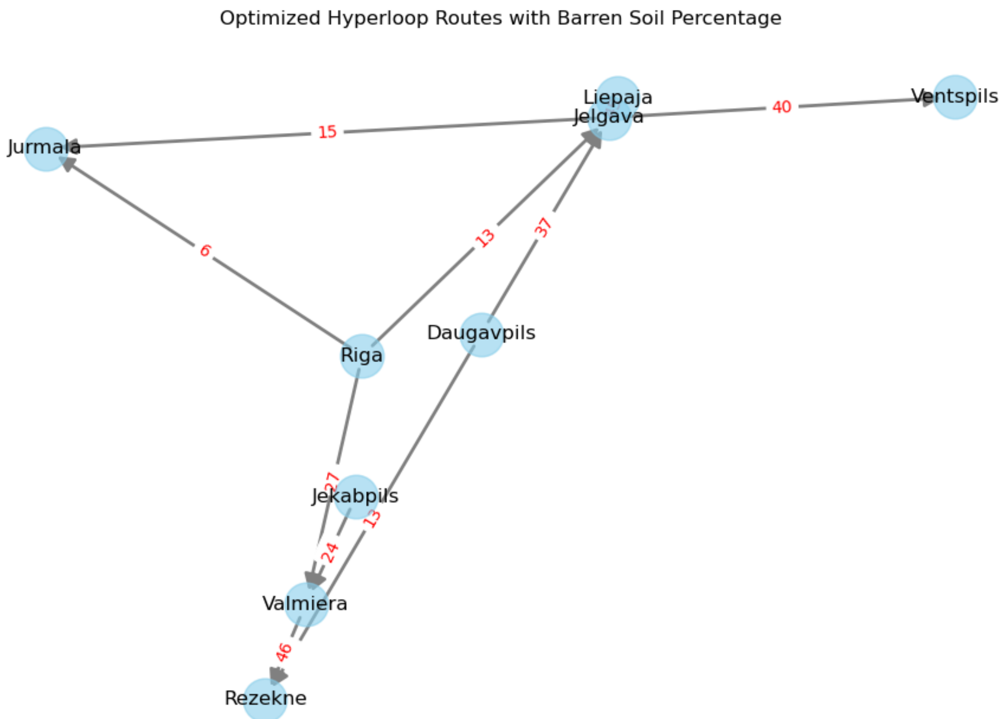
```
9   Jekabpils   Valmiera          24       101
```

[8]:
```python
import pulp
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd

# Optimized considering barren soils

prob = pulp.LpProblem("Hyperloop_Route_Optimization", pulp.LpMaximize)


data = {
    'From': ['Riga', 'Riga', 'Riga', 'Daugavpils', 'Daugavpils', 'Liepaja',␣
 ↪'Jelgava', 'Jelgava', 'Valmiera', 'Jekabpils'],
    'To': ['Jelgava', 'Jurmala', 'Valmiera', 'Jelgava', 'Rezekne', 'Jelgava',␣
 ↪'Jurmala', 'Ventspils', 'Rezekne', 'Valmiera'],
    'Barren soil': [13, 6, 27, 37, 13, 45, 15, 40, 46, 24],
    'Distance': [45, 38, 107, 230, 89, 182, 45, 176, 185, 101],
    'Selected': [1, 0, 1, 0, 1, 0, 1, 1, 0, 1]
}


df = pd.DataFrame(data)


routes = [(row['From'], row['To']) for _, row in df.iterrows()]
x = pulp.LpVariable.dicts("Route", routes, cat='Binary')

prob += pulp.lpSum([row['Distance'] * row['Barren soil'] / 100 *␣
 ↪x[(row['From'], row['To'])] for _, row in df.iterrows()])


cities = set(df['From']).union(set(df['To']))
for city in cities:
    prob += pulp.lpSum([x[(row['From'], row['To'])] for _, row in df.iterrows()␣
 ↪if row['From'] == city or row['To'] == city]) >= 1


prob.solve()


optimized_routes = []
for v in prob.variables():
    if v.varValue > 0:
        i, j = v.name.replace("Route_", "").split("_")
        optimized_routes.append((i, j))
```

```python
print("Status:", pulp.LpStatus[prob.status])
print("Total barren soil coverage:", pulp.value(prob.objective))
print("Optimized Hyperloop Routes:")
print(optimized_routes)


df['Selected'] = df.apply(lambda row: x[(row['From'], row['To'])].varValue,␣
 ↪axis=1)
selected_routes = df[df['Selected'] == 1]


G = nx.DiGraph()


for _, row in selected_routes.iterrows():
    G.add_edge(row['From'], row['To'], weight=row['Barren soil'])


pos = nx.spring_layout(G)


plt.figure(figsize=(12, 8))
nx.draw_networkx_nodes(G, pos, node_size=700, node_color='skyblue', alpha=0.6)


nx.draw_networkx_edges(G, pos, arrowstyle='-|>', arrowsize=20,␣
 ↪edge_color='gray', width=2)


nx.draw_networkx_labels(G, pos, font_size=12, font_family='sans-serif')


edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red')

plt.title('Optimized Hyperloop Routes with Barren Soil Percentage')
plt.axis('off')
plt.show()
```

```
Status: Optimal
Total barren soil coverage: 402.08000000000004
Optimized Hyperloop Routes:
[("('Daugavpils',", "'Jelgava')"), ("('Daugavpils',", "'Rezekne')"),
("('Jekabpils',", "'Valmiera')"), ("('Jelgava',", "'Jurmala')"), ("('Jelgava',",
"'Ventspils')"), ("('Liepaja',", "'Jelgava')"), ("('Riga',", "'Jelgava')"),
("('Riga',", "'Jurmala')"), ("('Riga',", "'Valmiera')"), ("('Valmiera',",
```

```
"'Rezekne')")]
```

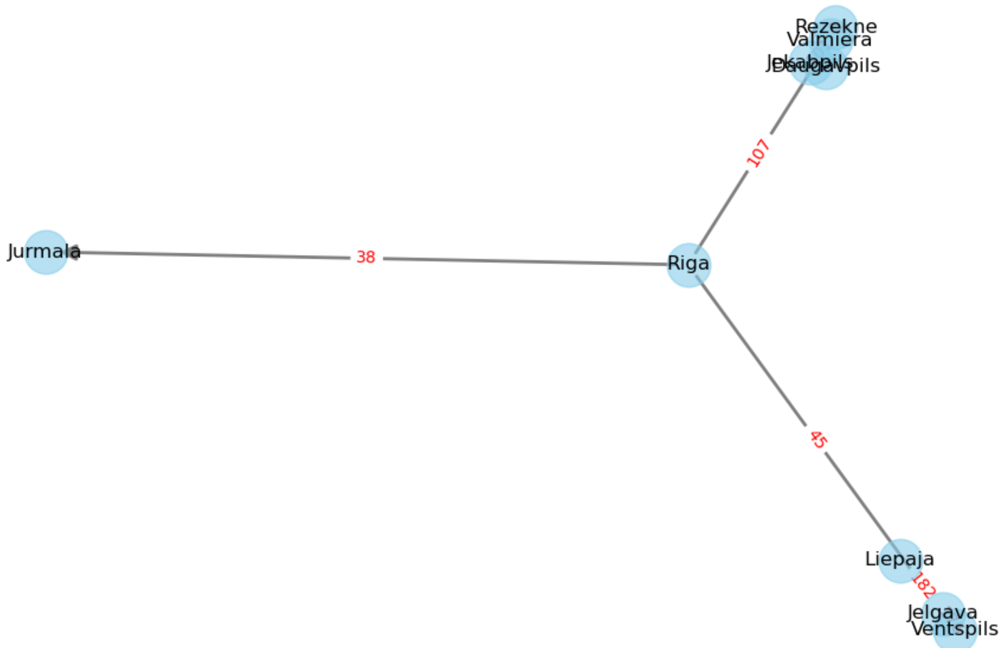



```
[10]: import pulp
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd

# Optimized without considering barren soils


data = {
    'From': ['Riga', 'Riga', 'Riga', 'Daugavpils', 'Daugavpils', 'Liepaja',
  'Jelgava', 'Jelgava', 'Valmiera', 'Jekabpils'],
    'To': ['Jelgava', 'Jurmala', 'Valmiera', 'Jelgava', 'Rezekne', 'Jelgava',
  'Jurmala', 'Ventspils', 'Rezekne', 'Valmiera'],
    'Barren soil': [13, 6, 27, 37, 13, 45, 15, 40, 46, 24],
    'Distance': [45, 38, 107, 230, 89, 182, 45, 176, 185, 101],
    'Selected': [1, 0, 1, 0, 1, 0, 1, 1, 0, 1]
}


df = pd.DataFrame(data)
```

```
routes = [(row['From'], row['To']) for _, row in df.iterrows()]
x = pulp.LpVariable.dicts("Route", routes, cat='Binary')


G = nx.Graph()
for _, row in df.iterrows():
    G.add_edge(row['From'], row['To'], weight=row['Distance'])


mst = nx.minimum_spanning_tree(G, weight='weight')


prob = pulp.LpProblem("Hyperloop_Route_Optimization", pulp.LpMinimize)


prob += pulp.lpSum([row['Distance'] * x[(row['From'], row['To'])] for _, row in␣
 ↪df.iterrows()])

for u, v in mst.edges():
    if (u, v) in x:
        prob += x[(u, v)] == 1
    elif (v, u) in x:
        prob += x[(v, u)] == 1


prob.solve()


optimized_routes = []
for v in prob.variables():
    if v.varValue > 0:
        i, j = v.name.replace("Route_", "").split("_")
        optimized_routes.append((i, j))

print("Status:", pulp.LpStatus[prob.status])
print("Total distance:", pulp.value(prob.objective))
print("Optimized Hyperloop Routes:")
print(optimized_routes)


df['Selected'] = df.apply(lambda row: x[(row['From'], row['To'])].varValue,␣
 ↪axis=1)
selected_routes = df[df['Selected'] == 1]
```

```python
G_optimized = nx.DiGraph()

for _, row in selected_routes.iterrows():
    G_optimized.add_edge(row['From'], row['To'], weight=row['Distance'])


pos = nx.spring_layout(G_optimized)


plt.figure(figsize=(12, 8))
nx.draw_networkx_nodes(G_optimized, pos, node_size=700, node_color='skyblue',␣
 ↪alpha=0.6)


nx.draw_networkx_edges(G_optimized, pos, arrowstyle='-|>', arrowsize=20,␣
 ↪edge_color='gray', width=2)


nx.draw_networkx_labels(G_optimized, pos, font_size=12,␣
 ↪font_family='sans-serif')


edge_labels = nx.get_edge_attributes(G_optimized, 'weight')
nx.draw_networkx_edge_labels(G_optimized, pos, edge_labels=edge_labels,␣
 ↪font_color='red')

plt.axis('off')
plt.show()
```

```
Status: Optimal
Total distance: 923.0
Optimized Hyperloop Routes:
[("('Daugavpils',", "'Rezekne')"), ("('Jekabpils',", "'Valmiera')"),
("('Jelgava',", "'Ventspils')"), ("('Liepaja',", "'Jelgava')"), ("('Riga',",
"'Jelgava')"), ("('Riga',", "'Jurmala')"), ("('Riga',", "'Valmiera')"),
("('Valmiera',", "'Rezekne')")]
```


Rezekne
Valmiera
107
Jurmala
38
Riga
45
Liepaja
182
Jelgava
Ventspils