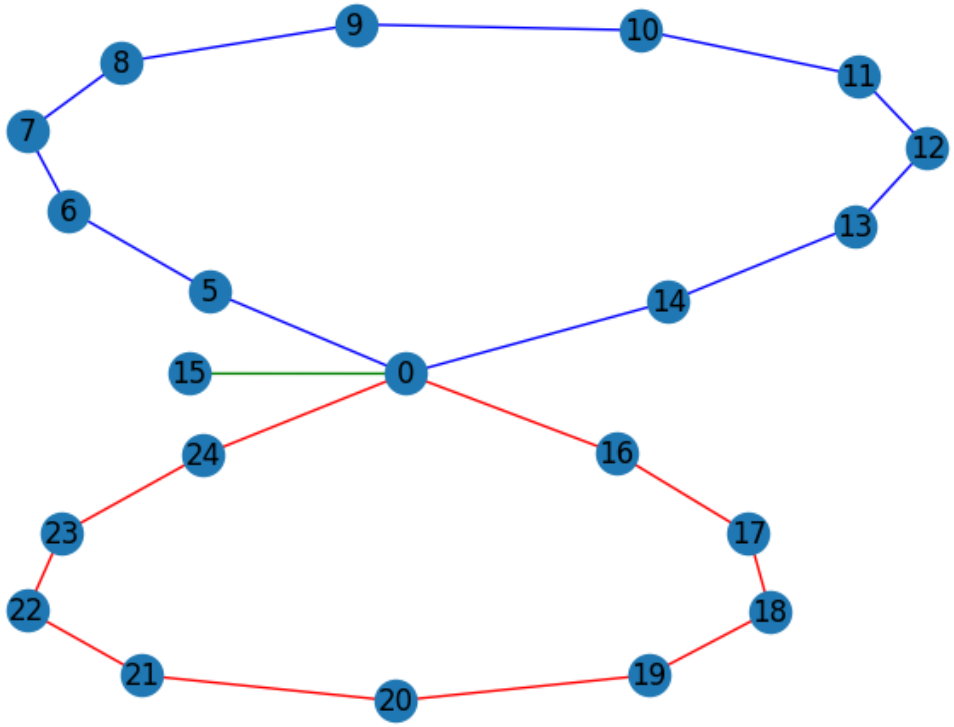
soluionPartitioningsolver1

March 24, 2023

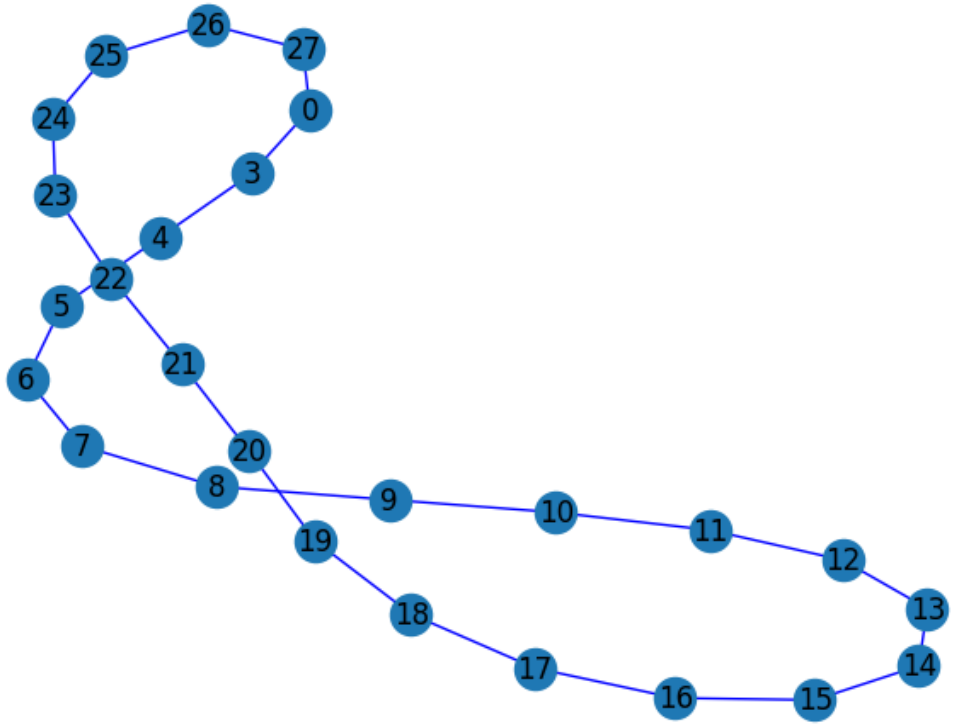
[1]:
```
colors = ['b', 'g', 'r']
```

[2]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Define the number of nodes in the graph
num_nodes = 25

# Define the VRP solver results
solution = [[0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 0], [0, 15, 0], [0, 16, 17,␣
 ↪18, 19, 20, 21, 22, 23, 24, 0]]

# Create a graph with only the edges that are part of the solution
G = nx.Graph()
for i, route in enumerate(solution):
    for j in range(len(route) - 1):
        node1 = route[j]
        node2 = route[j + 1]
        G.add_edge(node1, node2, color=colors[i])

# Visualize the graph with the routes
pos = nx.spring_layout(G)
edges = G.edges()
colors = [G[u][v]['color'] for u,v in edges]
nx.draw(G, pos=pos, edgelist=edges, edge_color=colors, with_labels=True)
plt.show()
```



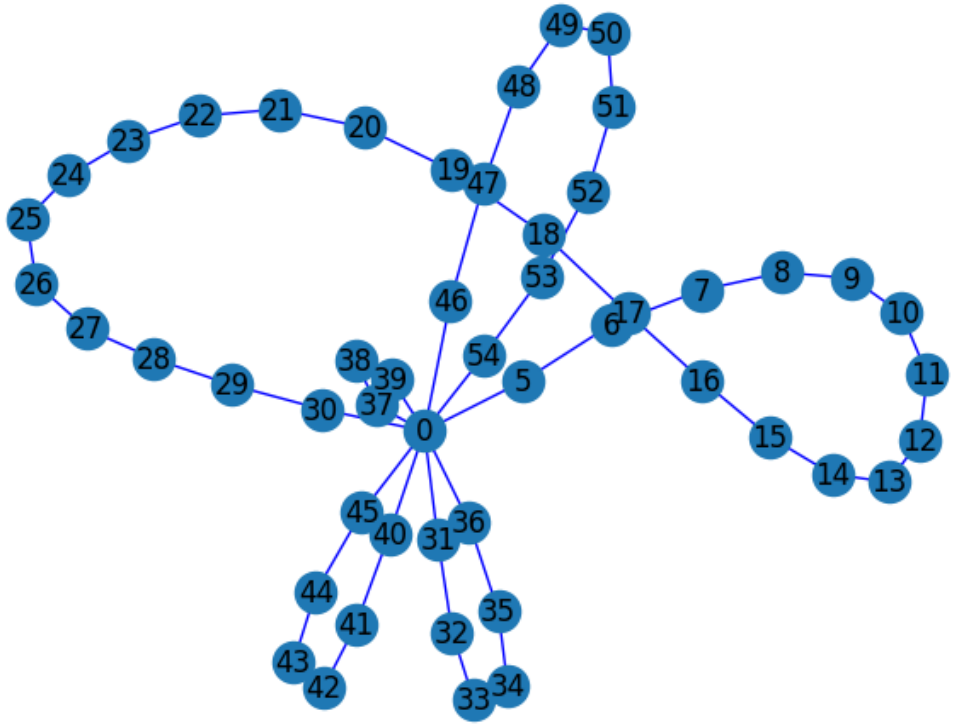
[3]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Define the number of nodes in the graph
num_nodes = 28

# Define the VRP solver results
solution = [[], [0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,␣
 ↪19, 20, 21, 22, 23, 24, 25, 26, 27, 0], [], [], []]

# Create a graph with only the edges that are part of the solution
G = nx.Graph()
for i, route in enumerate(solution):
    for j in range(len(route) - 1):
        node1 = route[j]
        node2 = route[j + 1]
        G.add_edge(node1, node2, color=colors[i])

# Visualize the graph with the routes
pos = nx.spring_layout(G)
```

```
edges = G.edges()
colors = [G[u][v]['color'] for u,v in edges]
nx.draw(G, pos=pos, edgelist=edges, edge_color=colors, with_labels=True)
plt.show()
```



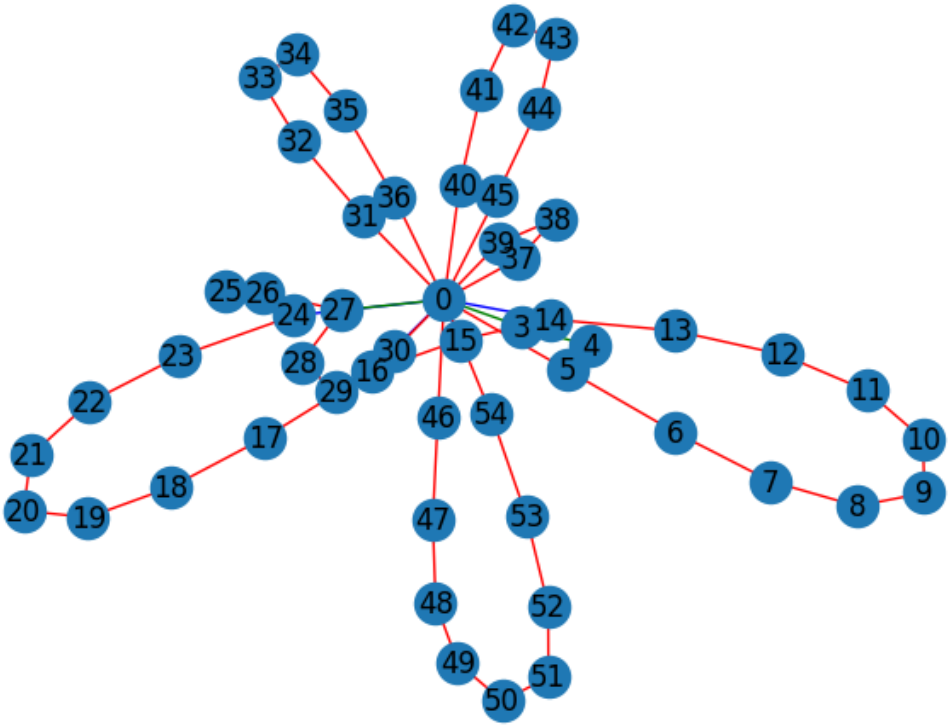
[4]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Define the number of nodes in the graph
num_nodes = 55

# Define the VRP solver results
solution = [[0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,␣
 ↪22, 23, 24, 25, 26, 27, 28, 29, 30, 0], [0, 40, 41, 42, 43, 44, 45, 0], [0,␣
 ↪31, 32, 33, 34, 35, 36, 0], [0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 0], [0,␣
 ↪37, 38, 39, 0]]

# Create a graph with only the edges that are part of the solution
G = nx.Graph()
for i, route in enumerate(solution):
```

```
    for j in range(len(route) - 1):
        node1 = route[j]
        node2 = route[j + 1]
        G.add_edge(node1, node2, color=colors[i])

# Visualize the graph with the routes
pos = nx.spring_layout(G)
edges = G.edges()
colors = [G[u][v]['color'] for u,v in edges]
nx.draw(G, pos=pos, edgelist=edges, edge_color=colors, with_labels=True)
plt.show()
```



[5]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Define the number of nodes in the graph
num_nodes = 55

# Define the VRP solver results
```

```
solution1 = [[0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 0], [0, 15, 0], [0, 16, 17,␣
 ↪18, 19, 20, 21, 22, 23, 24, 0]]
solution2 = [[], [0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,␣
 ↪19, 20, 21, 22, 23, 24, 25, 26, 27, 0], [], [], []]
solution3 = [[0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,␣
 ↪22, 23, 24, 25, 26, 27, 28, 29, 30, 0], [0, 40, 41, 42, 43, 44, 45, 0], [0,␣
 ↪31, 32, 33, 34, 35, 36, 0], [0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 0], [0,␣
 ↪37, 38, 39, 0]]

# Create a graph with only the edges that are part of the solutions
G = nx.Graph()
colors = ['b', 'g', 'r']*3
for i, solution in enumerate([solution1, solution2, solution3]):
    for j, route in enumerate(solution):
        for k in range(len(route) - 1):
            node1 = route[k]
            node2 = route[k + 1]
            G.add_edge(node1, node2, color=colors[i])

# Visualize the graph with the routes
pos = nx.spring_layout(G)
edges = G.edges()
colors = [G[u][v]['color'] for u,v in edges]
nx.draw(G, pos=pos, edgelist=edges, edge_color=colors, with_labels=True)
plt.show()
```


42
43
34
33
41
44
35
32
40
45
36
31
38
39
37
25
26
24
27
0
3
14
13
12
11
10
9
8
7
6
5
4
15
30
16
28
29
23
22
21
20
19
18
17
46
54
47
53
48
52
49
51
50