Mathematics for Automation and Robotics
MUAR 2023-24

Assignment 1: Graphs Problems

INDUSTRIALES
ETSII | UPM

Josep María Barberá Civera

Objectives

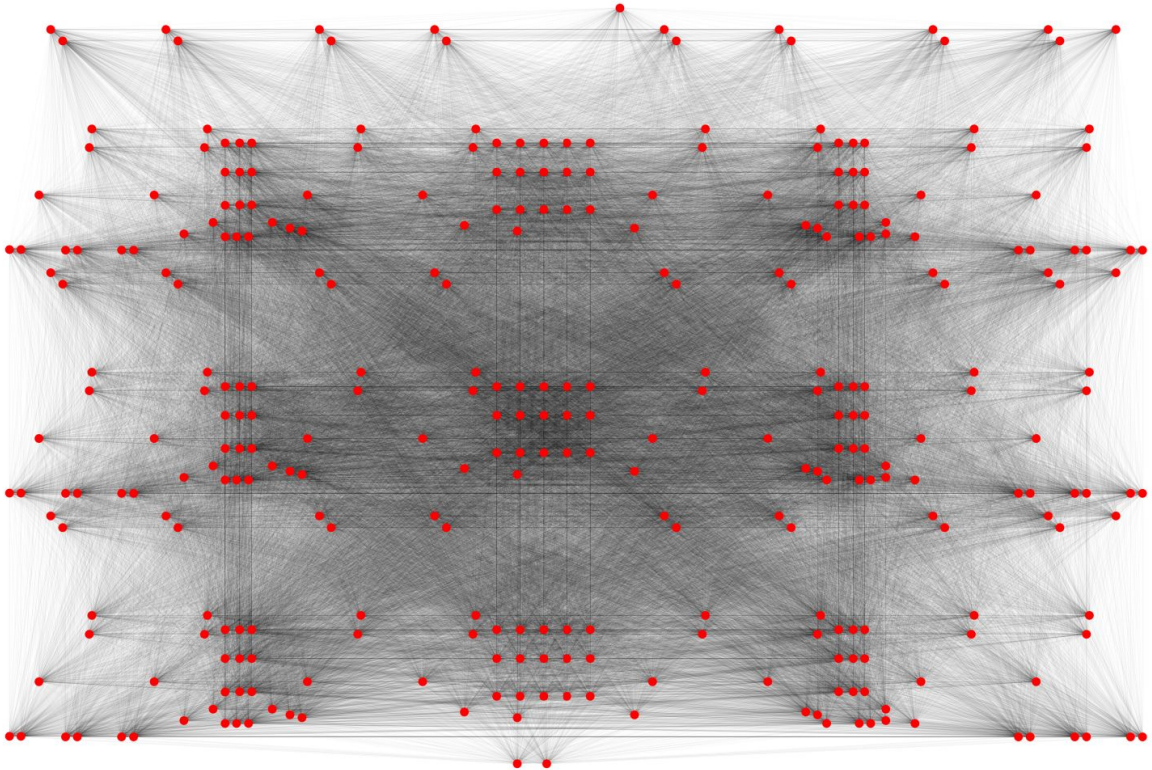
Algorithm development for graph problems

Graphs	A. lin318.tsp (undirected, weighted) B. kroA100.tsp (undirected, weighted) C. pcb442.tsp (undirected, weighted) D. bitcoins.csv (undirected, weighted)
Classic problems to be solved	1. Shortest Cycle (TSP problem), Chinese Postman and Shortest Path from node 1 to last node 1 to the last node (all graphs except D). 2. Minimum Spanning Tree (only to D).

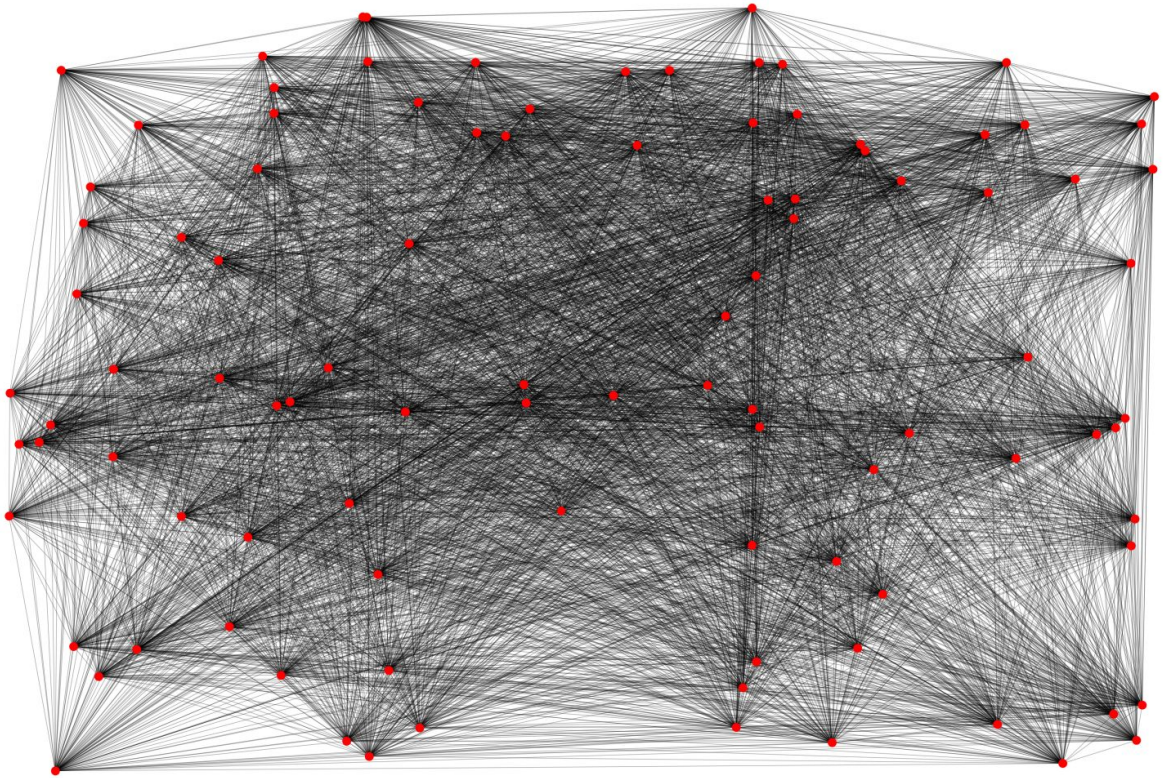
We are going to use the **NetworkX** python library for the plotting and the problem solution

A

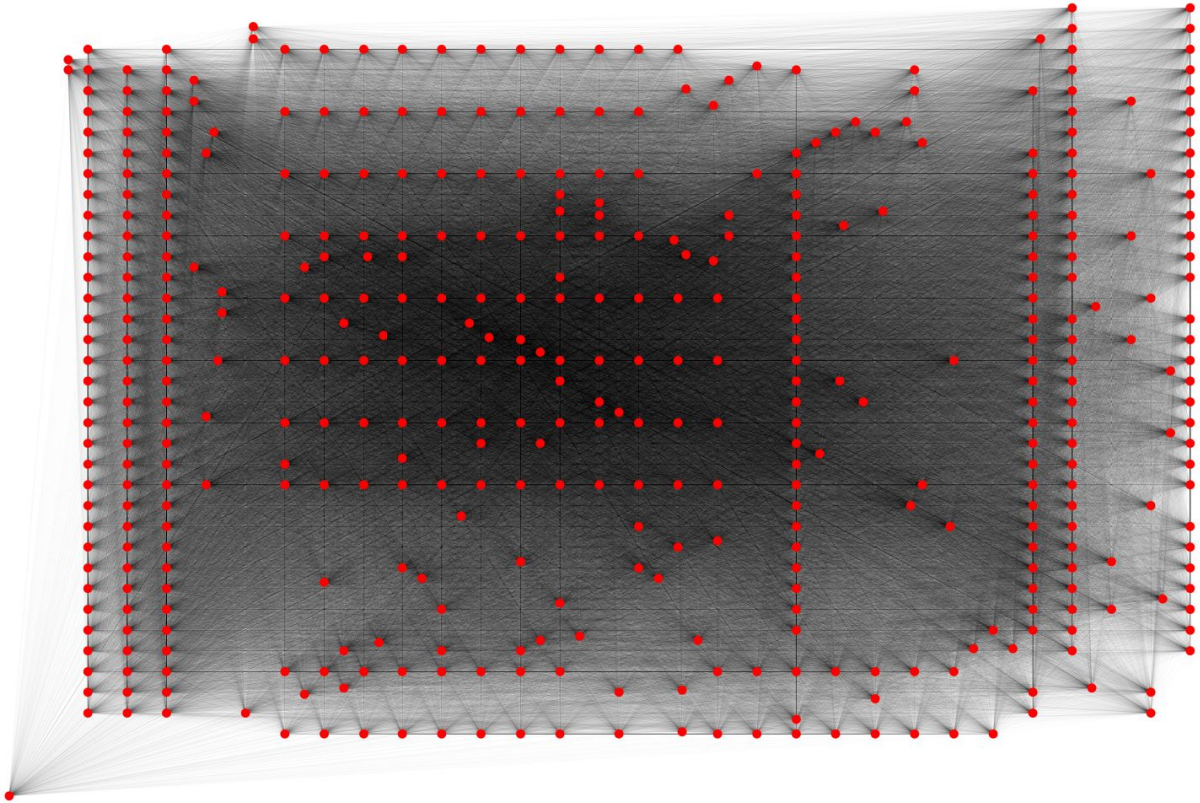
lin318.tsp (undirected, weighted)



B kroA100.tsp (undirected, weighted)



c pcb442.tsp (undirected, weighted)



TSP

```
import networkx as nx
import os
import math
import glob
import time
import matplotlib.pyplot as plt
import numpy as np
```

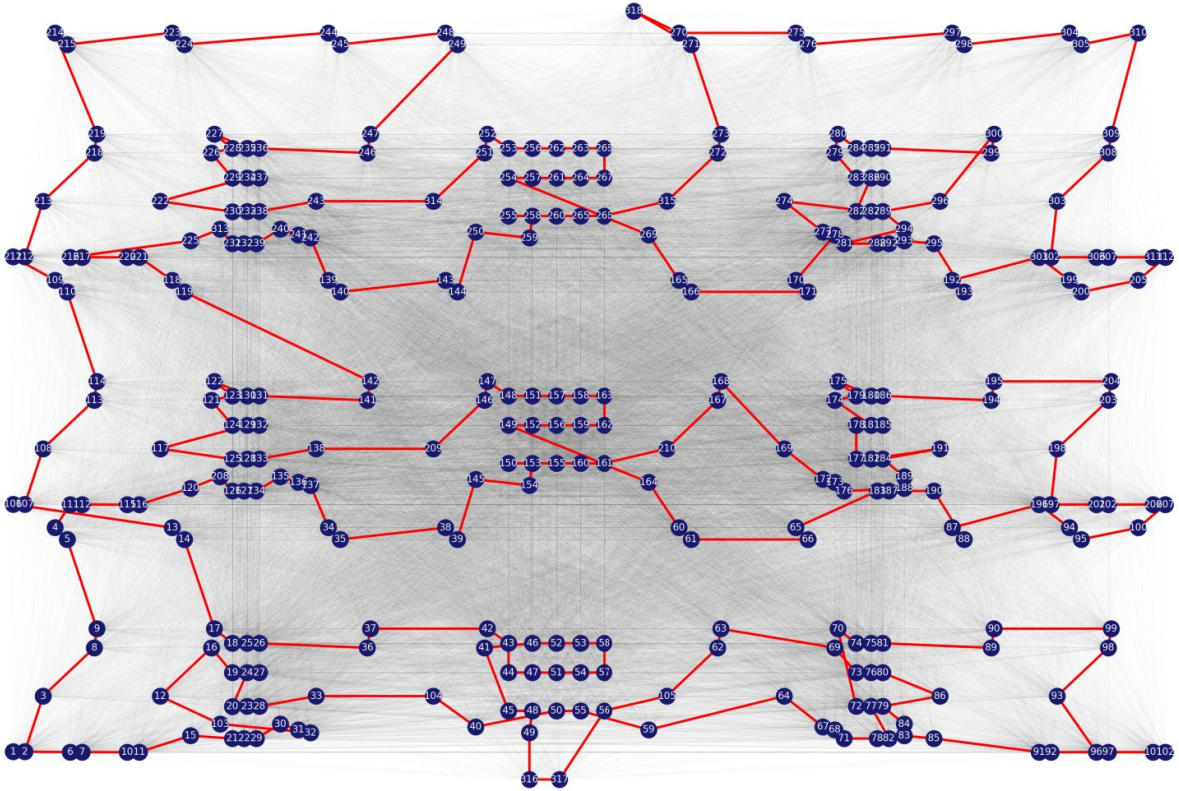
```
GRAPH = 1
if GRAPH == 1:
    ##############################################
    # TSP (Travellin Salesman Problem)
    ##############################################
    tsp = nx.approximation.traveling_salesman_problem
    path = tsp(G)
    plot_name = "tsp_" + file[:-4] + ".png"
elif GRAPH == 2:
    ##############################################
    # CPP or EULERIAN CIRCUIT = CPP (Chinese Postman Problem) (becuase we use full connected graphs)
    ##############################################
    if nx.is_eulerian(G):
            path = [u for u, v in nx.eulerian_circuit(G, source=1)]
    else:
        H = nx.eulerize(G)
        path = [u for u, v in nx.eulerian_path(H, source=1)]
    plot_name = "cpp_" + file[:-4] + ".png"
elif GRAPH == 3:
    ##############################################
    # SHORTEST PATH
    ##############################################
    path = nx.shortest_path(G, source=1, target=len(pos))
    plot_name = "shortest_path_" + file[:-4] + ".png"
    ##############################################
```

TSP

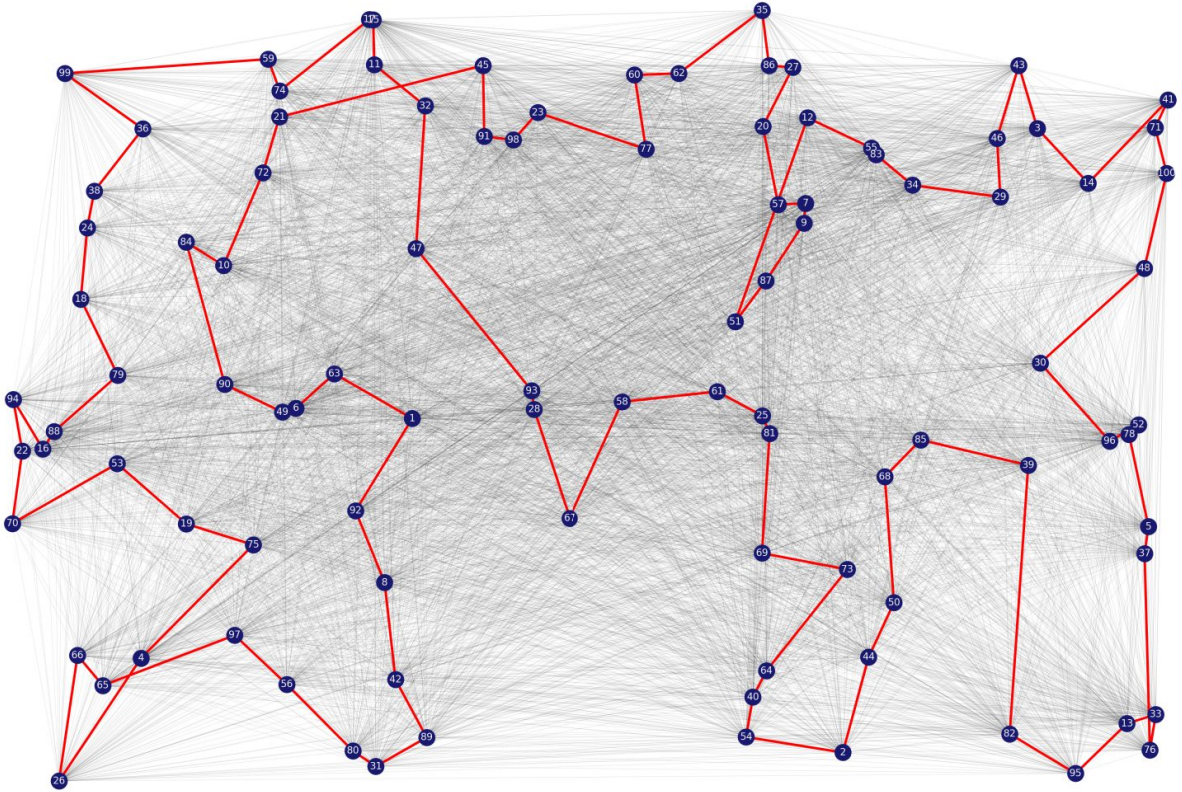
```
Graph problem took 43.6210 seconds
Here the path:
[1, 2, 3, 8, 9, 5, 4, 111, 112, 115, 116, 120, 208, 126, 127, 134, 135, 136, 137, 34, 35, 38, 39, 145, 154, 153,
150, 155, 160, 161, 210, 167, 168, 169, 172, 173, 176, 188, 189, 184, 191, 182, 177, 178, 185, 181, 174, 179, 17
5, 180, 186, 194, 195, 204, 203, 198, 197, 201, 202, 206, 207, 100, 95, 94, 196, 87, 88, 190, 187, 183, 65, 66,
61, 60, 164, 149, 152, 156, 159, 162, 163, 158, 157, 151, 148, 147, 146, 209, 138, 133, 128, 125, 117, 129, 132,
124, 121, 123, 122, 130, 131, 141, 142, 119, 118, 221, 220, 217, 216, 225, 313, 231, 232, 239, 240, 241, 242, 13
9, 140, 143, 144, 250, 259, 258, 255, 260, 265, 266, 315, 272, 273, 271, 318, 270, 275, 276, 297, 298, 304, 305,
310, 309, 308, 303, 302, 306, 307, 311, 312, 205, 200, 199, 301, 192, 193, 295, 293, 292, 288, 281, 294, 289, 29
6, 300, 299, 291, 285, 284, 280, 279, 283, 290, 286, 282, 287, 274, 277, 278, 170, 171, 166, 165, 269, 254, 257,
261, 264, 267, 268, 263, 262, 256, 253, 252, 251, 314, 243, 238, 233, 230, 222, 234, 237, 229, 226, 228, 227, 23
5, 236, 246, 247, 249, 248, 245, 244, 224, 223, 215, 214, 219, 218, 213, 212, 211, 109, 110, 114, 113, 108, 107,
106, 13, 14, 17, 18, 25, 26, 36, 37, 42, 43, 44, 47, 51, 54, 57, 58, 53, 52, 46, 41, 45, 48, 49, 316, 317, 56, 1
05, 62, 63, 69, 73, 76, 80, 86, 79, 84, 83, 85, 91, 92, 102, 101, 97, 96, 93, 98, 99, 90, 89, 81, 75, 74, 70, 7
2, 77, 82, 78, 71, 68, 67, 64, 59, 55, 50, 40, 104, 33, 28, 23, 20, 24, 27, 19, 16, 12, 103, 32, 31, 30, 29, 22,
21, 15, 11, 10, 7, 6, 1]
With a total length of 47464.57 distance units
```

```
Graph problem took 1.2782 seconds
Here the path:
[1, 92, 8, 42, 89, 31, 80, 56, 97, 65, 66, 26, 4, 75, 19, 53, 70, 22, 94, 16, 88, 79, 18, 24, 38, 36, 99, 59, 7
4, 17, 15, 11, 32, 47, 93, 28, 67, 58, 61, 25, 81, 69, 73, 64, 40, 54, 2, 44, 50, 68, 85, 39, 82, 95, 13, 33, 7
6, 37, 5, 78, 52, 96, 30, 48, 100, 71, 41, 14, 3, 43, 46, 29, 34, 83, 55, 12, 51, 87, 9, 7, 57, 20, 27, 86, 35,
62, 60, 77, 23, 98, 91, 45, 21, 72, 10, 84, 90, 49, 6, 63, 1]
With a total length of 23292.98 distance units
```

A TSP



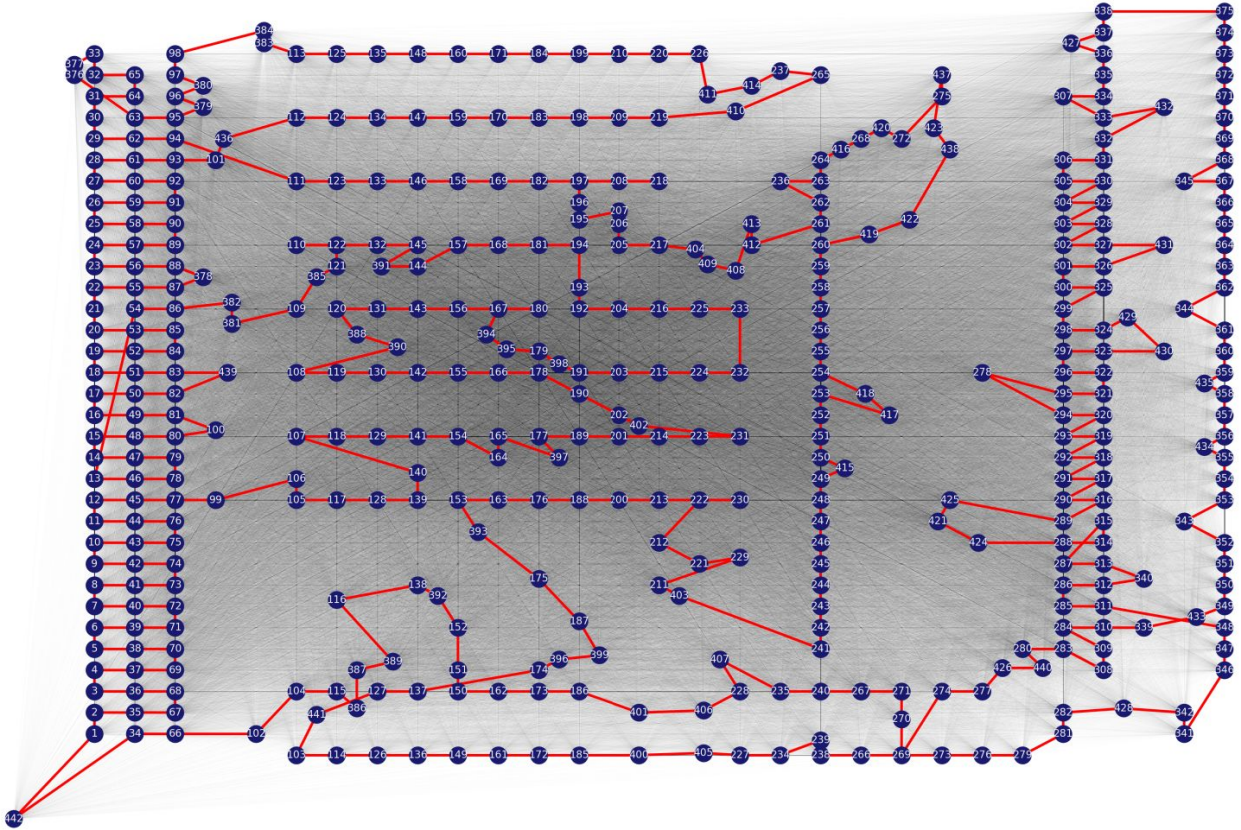
TSP


115
35
59
11
45
60
62
86
27
43
99
74
21
32
23
91
98
77
20
12
46
3
41
71
36
72
38
24
84
10
47
34
29
14
100
57
7
9
87
51
48
18
79
90
63
49
6
1
93
28
58
61
25
81
30
94
88
22
16
53
70
19
75
92
67
8
97
66
4
65
56
42
80
31
89
26
69
73
50
44
64
40
54
2
85
68
39
52
78
96
5
37
13
33
82
95
76

TSP

```
Graph problem took 123.3416 seconds
Here the path:
[1, 442, 34, 66, 102, 104, 115, 386, 387, 389, 116, 138, 392, 152, 151, 150, 162, 173, 186, 401, 406, 228, 407,
235, 240, 267, 271, 270, 269, 274, 277, 426, 440, 280, 283, 308, 309, 284, 310, 339, 433, 349, 350, 351, 352, 34
3, 353, 354, 355, 434, 356, 357, 358, 435, 359, 360, 361, 344, 362, 363, 364, 365, 366, 367, 345, 368, 369, 370,
371, 372, 373, 374, 375, 338, 337, 427, 336, 335, 334, 307, 333, 432, 332, 331, 306, 305, 330, 304, 329, 303, 32
8, 302, 327, 431, 326, 301, 300, 325, 299, 298, 324, 429, 430, 323, 297, 296, 322, 321, 295, 278, 294, 320, 293,
319, 292, 318, 291, 317, 290, 316, 289, 425, 421, 424, 288, 314, 315, 287, 313, 340, 312, 286, 285, 311, 348, 34
7, 346, 341, 342, 428, 282, 281, 279, 276, 273, 266, 238, 239, 234, 227, 405, 400, 185, 172, 161, 149, 136, 126,
114, 103, 441, 127, 137, 174, 396, 399, 187, 175, 393, 153, 163, 176, 188, 200, 213, 230, 222, 212, 221, 229, 21
1, 403, 241, 242, 243, 244, 245, 246, 247, 248, 249, 415, 250, 251, 252, 253, 417, 418, 254, 255, 256, 257, 258,
259, 260, 419, 422, 438, 423, 437, 275, 272, 420, 268, 416, 264, 263, 236, 262, 261, 412, 413, 408, 409, 404, 21
7, 205, 206, 207, 195, 196, 197, 218, 208, 182, 169, 158, 146, 133, 123, 111, 94, 62, 29, 30, 31, 64, 65, 32, 3
3, 377, 376, 63, 95, 379, 96, 380, 97, 98, 384, 383, 113, 125, 135, 148, 160, 171, 184, 199, 210, 220, 226, 411,
414, 237, 265, 410, 219, 209, 198, 183, 170, 159, 147, 134, 124, 112, 436, 101, 93, 61, 28, 27, 60, 92, 91, 59,
26, 25, 58, 90, 89, 57, 24, 23, 56, 88, 378, 87, 55, 22, 21, 20, 53, 85, 84, 52, 19, 18, 51, 83, 439, 82, 50, 1
7, 16, 49, 81, 100, 80, 48, 15, 14, 47, 79, 78, 46, 13, 54, 86, 382, 381, 109, 385, 121, 122, 110, 132, 145, 39
1, 144, 157, 168, 181, 194, 193, 192, 204, 216, 225, 233, 232, 224, 215, 203, 191, 398, 179, 395, 394, 167, 180,
156, 143, 131, 120, 388, 390, 108, 119, 130, 142, 155, 166, 178, 190, 202, 402, 231, 223, 214, 201, 189, 177, 39
7, 165, 164, 154, 141, 129, 118, 107, 140, 139, 128, 117, 105, 106, 99, 77, 45, 12, 11, 44, 76, 75, 43, 10, 9, 4
2, 74, 73, 41, 8, 7, 40, 72, 71, 39, 6, 5, 38, 70, 69, 37, 4, 3, 36, 68, 67, 35, 2, 1]
With a total length of 54871.72 distance units
```

TSP



CPP

1
```
Graph: lin318.tsp
Graph problem took 22.2826 seconds
Here the path:
[1, 2, 3, 1, 4, 2, 5, 1, 6, 2, 7, 1, 8, 2, 9, 1, 10, 2, 11, 1, 12, 2, 13, 1, 14, 2, 15, 1, 16, 2, 17, 1, 18, 2,
19, 1, 20, 2, 21, 1]
With a total length of 93601595.26 distance units
With a total distance of graph: 93197470.43491593
```

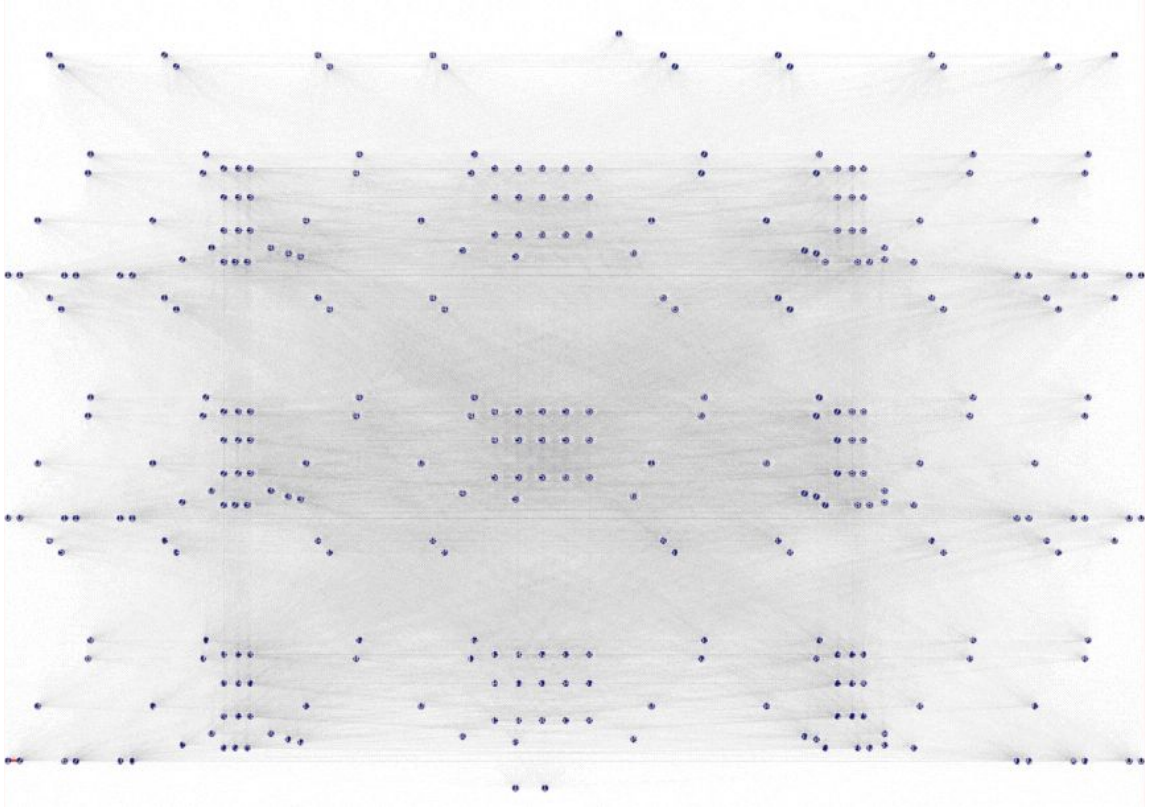
2
```
Graph: kroA100.tsp
Graph problem took 1.8073 seconds
Here the path:
[1, 2, 3, 1, 4, 2, 5, 1, 6, 2, 7, 1, 8, 2, 9, 1, 10, 2, 11, 1, 12, 2, 13, 1, 14, 2, 15, 1, 16, 2, 17, 1, 18, 2,
19, 1, 20, 2, 21, 1]
With a total length of 8548179.00 distance units
With a total distance of graph: 8467999.368139254
```

3
```
Graph: pcb442.tsp
Graph problem took 79.7362 seconds
Here the path:
[1, 2, 3, 1, 4, 2, 5, 1, 6, 2, 7, 1, 8, 2, 9, 1, 10, 2, 11, 1, 12, 2, 13, 1, 14, 2, 15, 1, 16, 2, 17, 1, 18, 2,
19, 1, 20, 2, 21, 1]
With a total length of 170803416.36 distance units
With a total distance of graph: 170361591.62289953
```

CPP

```
Graph: kroA100.tsp
Graph problem took 0.9095 seconds
Here the path:
[1, 2, 3, 1, 4, 2, 5, 1, 6, 2, 7, 1, 8, 2, 9, 1, 10, 2, 11, 1, 12, 2, 13, 1, 14, 2, 15, 1, 16, 2, 17, 1, 18, 2,
19, 1, 20, 2, 21, 1, 22, 2, 23, 1, 24, 2, 25, 1, 26, 2, 27, 1, 28, 2, 29, 1, 30, 2, 31, 1, 32, 2, 33, 1, 34, 2,
35, 1, 36, 2, 37, 1, 38, 2, 39, 1, 40, 2, 41, 1, 42, 2, 43, 1, 44, 2, 45, 1, 46, 2, 47, 1, 48, 2, 49, 1, 50, 2,
51, 1, 52, 2, 53, 1, 54, 2, 55, 1, 56, 2, 57, 1, 58, 2, 59, 1, 60, 2, 61, 1, 62, 2, 63, 1, 64, 2, 65, 1, 66, 2,
67, 1, 68, 2, 69, 1, 70, 2, 71, 1, 72, 2, 73, 1, 74, 2, 75, 1, 76, 2, 77, 1, 78, 2, 79, 1, 80, 2, 81, 1, 82, 2,
83, 1, 84, 2, 85, 1, 86, 2, 87, 1, 88, 2, 89, 1, 90, 2, 91, 1, 92, 2, 93, 1, 94, 2, 95, 1, 96, 2, 97, 1, 98, 2,
99, 1, 100, 2, 99, 3, 4, 5, 3, 6, 4, 7, 3, 8, 4, 9, 3, 10, 4, 11, 3, 12, 4, 13, 3, 14, 4, 15, 3, 16, 4, 17, 3,
18, 4, 19, 3, 20, 4, 21, 3, 22, 4, 23, 3, 24, 4, 25, 3, 26, 4, 27, 3, 28, 4, 29, 3, 30, 4, 31, 3, 32, 4, 33, 3,
34, 4, 35, 3, 36, 4, 37, 3, 38, 4, 39, 3, 40, 4, 41, 3, 42, 4, 43, 3, 44, 4, 45, 3, 46, 4, 47, 3, 48, 4, 49, 3,
50, 4, 51, 3, 52, 4, 53, 3, 54, 4, 55, 3, 56, 4, 57, 3, 58, 4, 59, 3, 60, 4, 61, 3, 62, 4, 63, 3, 64, 4, 65, 3,
66, 4, 67, 3, 68, 4, 69, 3, 70, 4, 71, 3, 72, 4, 73, 3, 74, 4, 75, 3, 76, 4, 77, 3, 78, 4, 79, 3, 80, 4, 81, 3,
82, 4, 83, 3, 84, 4, 85, 3, 86, 4, 87, 3, 88, 4, 89, 3, 90, 4, 91, 3, 92, 4, 93, 3, 94, 4, 95, 3, 96, 4, 97, 3,
98, 3, 100, 4, 97, 5, 6, 7, 5, 8, 6, 9, 5, 10, 6, 11, 5, 12, 6, 13, 5, 14, 6, 15, 5, 16, 6, 17, 5, 18, 6, 19,
5, 20, 6, 21, 5, 22, 6, 23, 5, 24, 6, 25, 5, 26, 6, 27, 5, 28, 6, 29, 5, 30, 6, 31, 5, 32, 6, 33, 5, 34, 6, 35,
5, 36, 6, 37, 5, 38, 6, 39, 5, 40, 6, 41, 5, 42, 6, 43, 5, 44, 6, 45, 5, 46, 6, 47, 5, 48, 6, 49, 5, 50, 6, 51,
5, 52, 6, 53, 5, 54, 6, 55, 5, 56, 6, 57, 5, 58, 6, 59, 5, 60, 6, 61, 5, 62, 6, 63, 5, 64, 6, 65, 5, 66, 6, 67,
```

A CPP



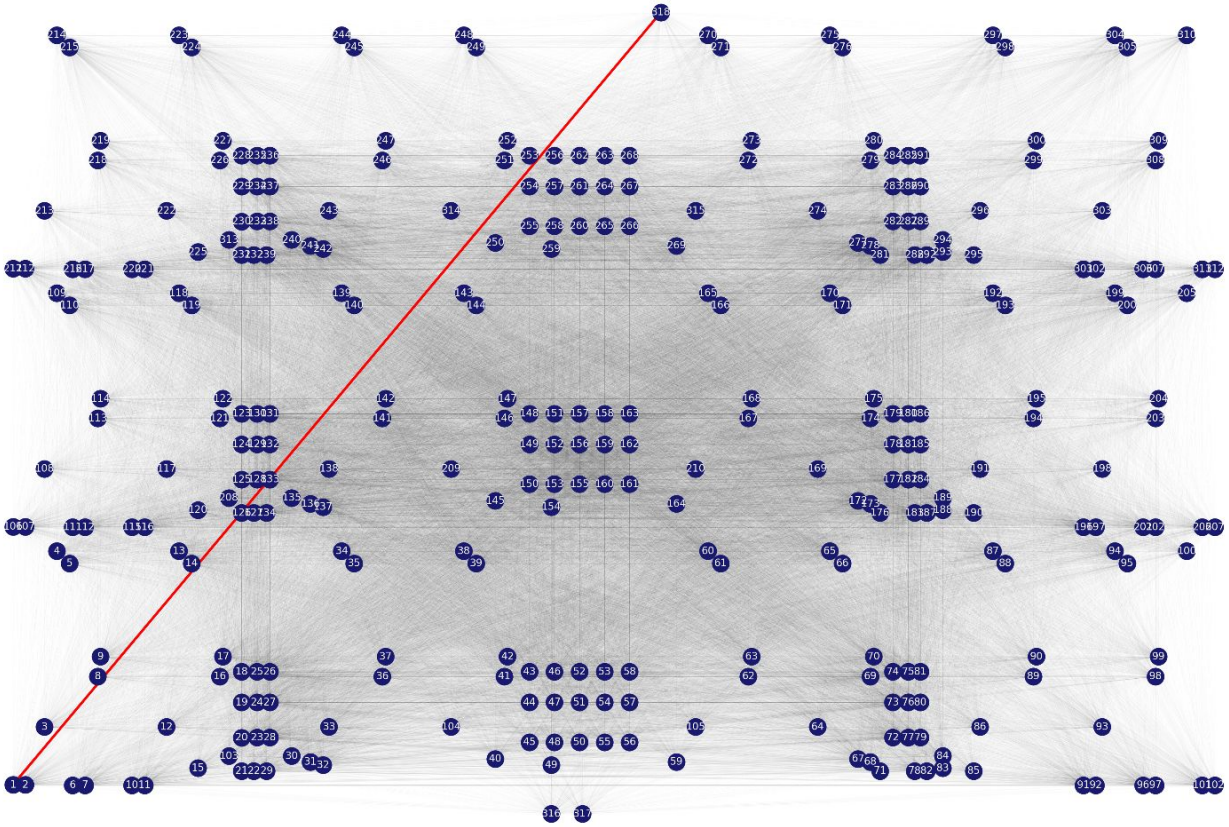
Only first 30 edges

Shortest Path (trivial?)

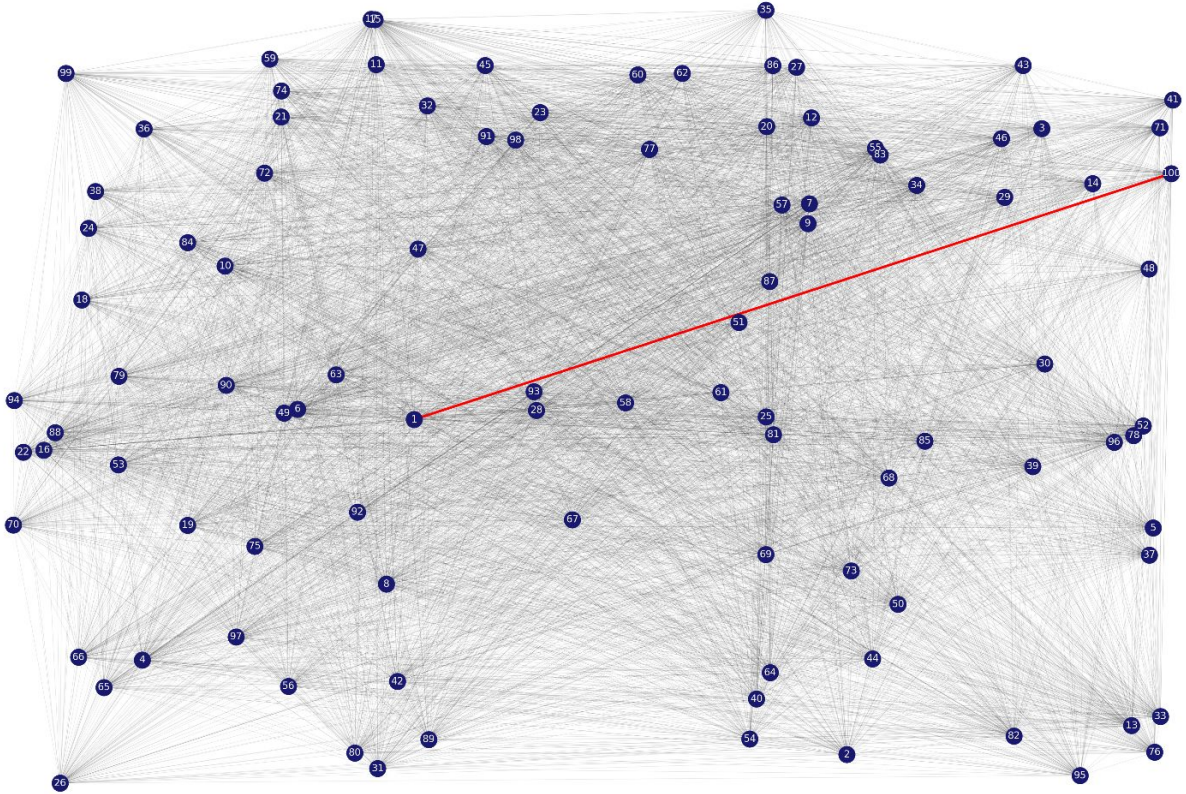
```
##############################################
# SHORTEST PATH
##############################################
path = nx.shortest_path(G, source=1, target=len(pos))
plot_name = "shortest_path_" + file[:-4] + ".png"
##############################################
end_time = time.time()

# Compute the total length for the path
length = compute_length(G, path)

print(f"Graph problem took {end_time - start_time:.4f} seconds")
print(f"Here the path:\n{path}")
print(f"With a total length of {length:.2f} distance units")

edge_list = list(nx.utils.pairwise(path))
plot_path(G, pos, edge_list, plot_name)
```

```
Graph problem took 0.0004 seconds
Here the path:
[1, 318]
With a total length of 4304.55 distance units
Graph problem took 0.0001 seconds
Here the path:
[1, 100]
With a total length of 2643.49 distance units
Graph problem took 0.0008 seconds
Here the path:
[1, 442]
With a total length of 447.21 distance units
```

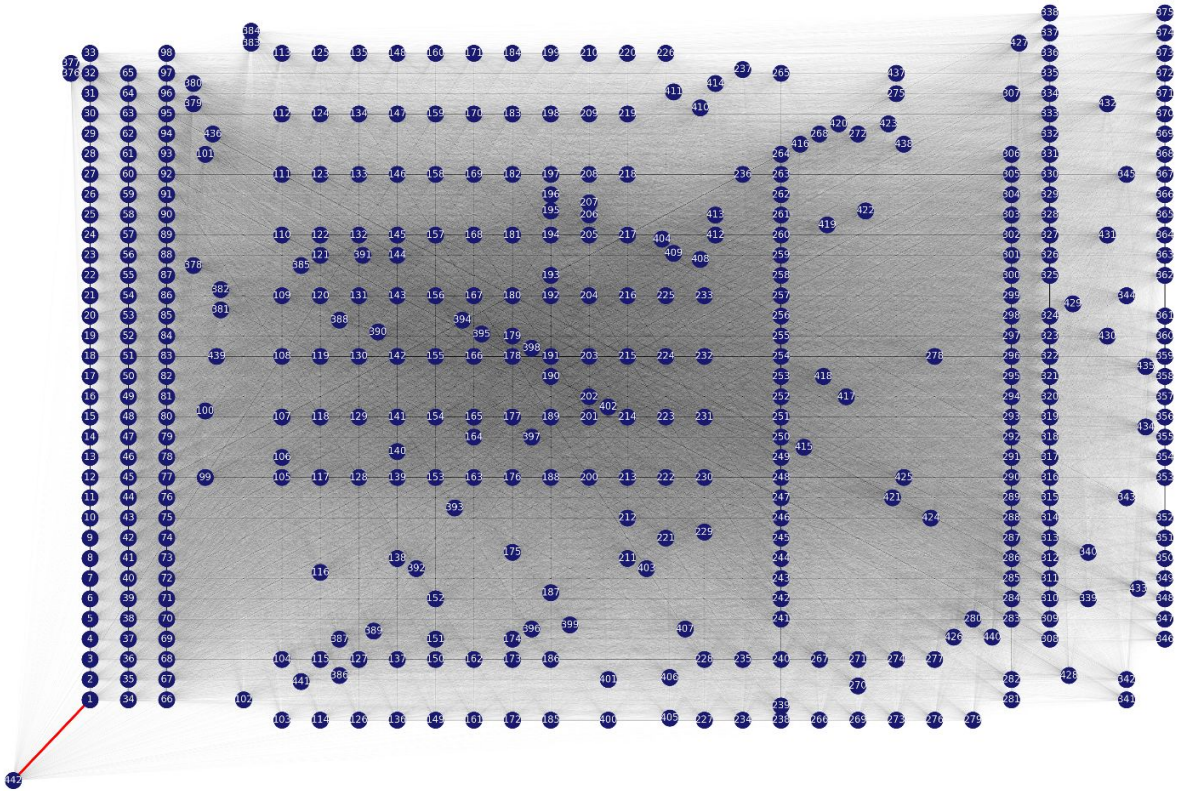
A Shortest Path



B Shortest Path

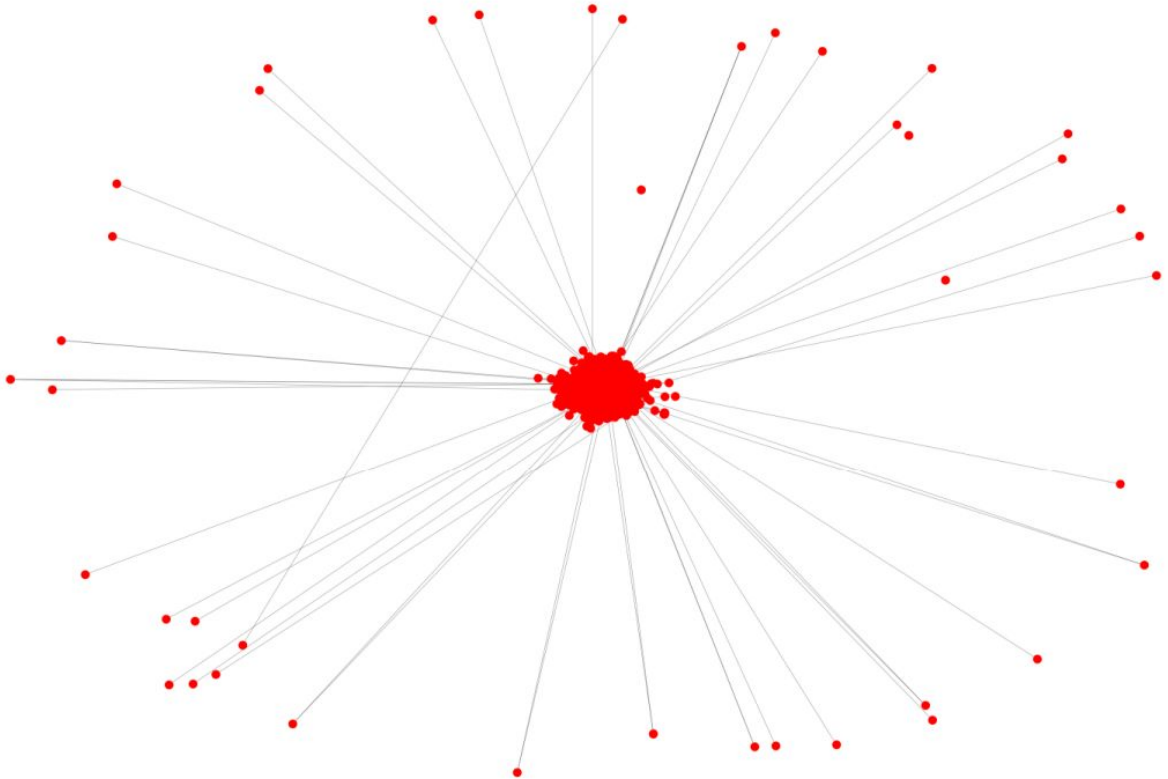


B Shortest Path

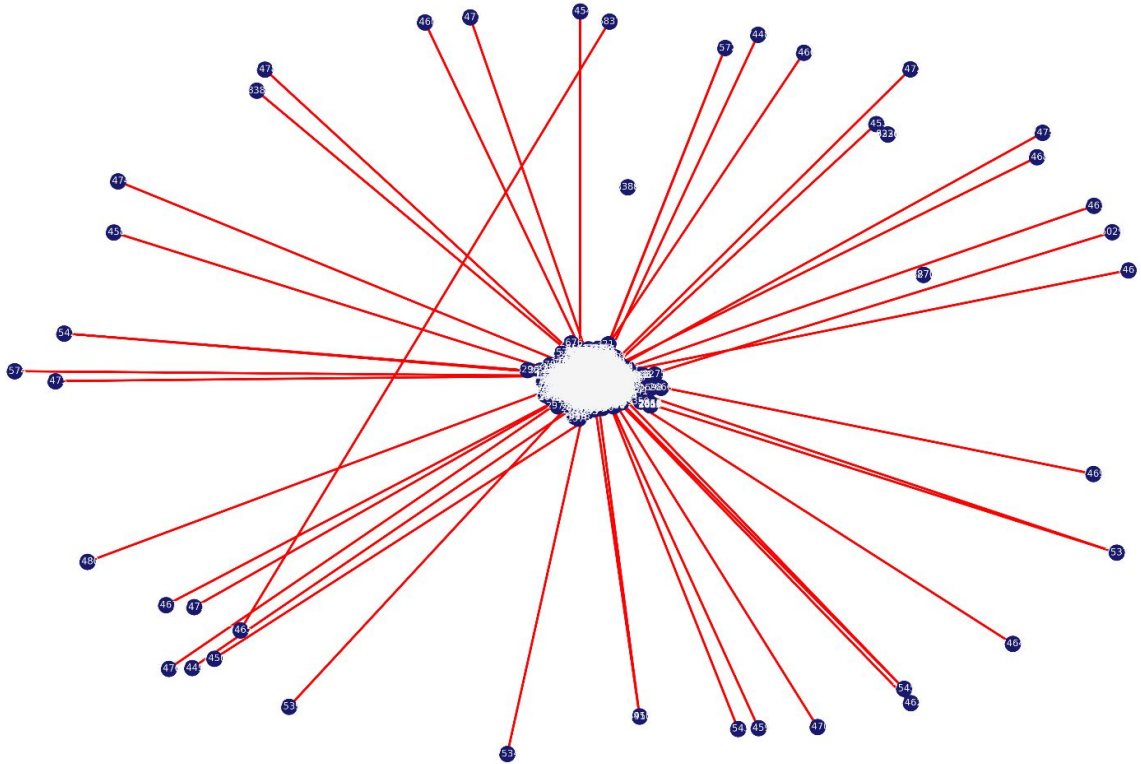


D

Bitcoins.csv



Minimum Spanning Tree



Minimum Spanning Tree

```
G = create_graph(tsp_files[0])
print(G)
pos = nx.spring_layout(G, seed=7)
plot_graph(G,pos, "bitcoins")
T = nx.minimum_spanning_tree(G, weight='weight')
edge_list = T.edges()
plot_path(G, pos, edge_list, "bitcoins")
```

```
Graph with 3783 nodes and 14124 edges
```

```
print(T)
print(T.edges())
```

```
Graph with 3783 nodes and 3778 edges
[(1, 7348), (1, 7425), (1, 1025), (1, 1029), (1, 1034), (1, 1051), (1, 1066), (1, 1072), (1, 1117), (1, 1128),
(1, 1129), (1, 1136), (1, 1148), (1, 115), (1, 1167), (1, 1186), (1, 1197), (1, 1232), (1, 1248), (1, 1251),
(1, 1282), (1, 1283), (1, 1284), (1, 1285), (1, 1288), (1, 1302), (1, 1309), (1, 1313), (1, 1318), (1, 1320),
(1, 1331), (1, 1336), (1, 1337), (1, 1339), (1, 1340), (1, 1342), (1, 1356), (1, 1365), (1, 1382), (1, 1413),
(1, 1436), (1, 1442), (1, 1457), (1, 1458), (1, 1465), (1, 1475), (1, 1483), (1, 1491), (1, 1492), (1, 1494),
(1, 1500), (1, 1505), (1, 1510), (1, 1520), (1, 1521), (1, 1525), (1, 1542), (1, 1574), (1, 158), (1, 1589),
(1, 1590), (1, 1708), (1, 1715), (1, 1720), (1, 1736), (1, 1742), (1, 1749), (1, 1750), (1, 178), (1, 1793),
(1, 1796), (1, 1822), (1, 1826), (1, 1835), (1, 1839), (1, 1840), (1, 1842), (1, 1843), (1, 1846), (1, 1847),
(1, 1862), (1, 1875), (1, 1885), (1, 1886), (1, 1900), (1, 1923), (1, 1942), (1, 2065), (1, 2090), (1, 2093),
(1, 2102), (1, 2108), (1, 2109), (1, 2117), (1, 2119), (1, 2127), (1, 2129), (1, 2136), (1, 2144), (1, 2147),
(1, 2152), (1, 2163), (1, 2166), (1, 2173), (1, 2176), (1, 2177), (1, 2178), (1, 2181), (1, 2185), (1, 2198),
(1, 2217), (1, 2235), (1, 2252), (1, 2261), (1, 2271), (1, 2273), (1, 2276), (1, 2281), (1, 2305), (1, 2315),
(1, 2324), (1, 2338), (1, 2342), (1, 2351), (1, 2352), (1, 2368), (1, 2375), (1, 2383), (1, 2391), (1, 2402),
(1, 2427), (1, 2448), (1, 2498), (1, 250), (1, 2559), (1, 2605), (1, 2619), (1, 2632), (1, 264), (1, 2687), (1,
2695), (1, 2700), (1, 2707), (1, 2713), (1, 2725), (1, 2731), (1, 2742), (1, 2743), (1, 2744), (1, 2746), (1, 2
747), (1, 2750), (1, 2752), (1, 2756), (1, 276), (1, 2763), (1, 2764), (1, 2767), (1, 2771), (1, 2772), (1, 277
5), (1, 2779), (1, 2780), (1, 2782), (1, 2783), (1, 2784), (1, 2786), (1, 2793), (1, 2795), (1, 2801), (1, 280
6), (1, 2810), (1, 2814), (1, 2820), (1, 2823), (1, 2826), (1, 2830), (1, 2837), (1, 2846), (1, 2848), (1, 285
8), (1, 2870), (1, 2881), (1, 2892), (1, 2898), (1, 290), (1, 2904), (1, 2907), (1, 2909), (1, 291), (1, 2929),
```

Minimum Spanning Tree

```
print(T)
MST_dim = 0
for edge in T.edges():
    u, v = edge
    w = float(T[u][v]["weight"])
    MST_dim = MST_dim + w
print(MST_dim)
```

```
Graph with 3783 nodes and 3778 edges
37234.0
```