traffic_simulator

February 16, 2024

[33]:

```
from datetime import datetime, timedelta
from pandas import DataFrame
import matplotlib.pyplot as plt
import networkx as nx
from networkx import Graph
from networkx.drawing.nx_agraph import write_dot, graphviz_layout
from networkx.readwrite import json_graph
import os
from random import randint
import sys
from typing import Tuple
```

[34]:

```
sys.path.append("../")
tests_dir = os.path.abspath("../tests")
sys.path.append(tests_dir)
```

[35]:

```
from traffic_simulator.city_map import CityMap
from traffic_simulator.model import TimeDeltaDiff
from traffic_simulator.traffic_analysis import TrafficAnalyzer
from traffic_simulator.traffic_simulation import Simulator
from conftest import generate_static_city_map, generate_static_trips
```

Generate City Map

[36]:

```
r2_city_map = generate_static_city_map()
r2_city_map
```

[36]:

```
<networkx.classes.graph.Graph at 0x12852c3d0>
```

[37]:

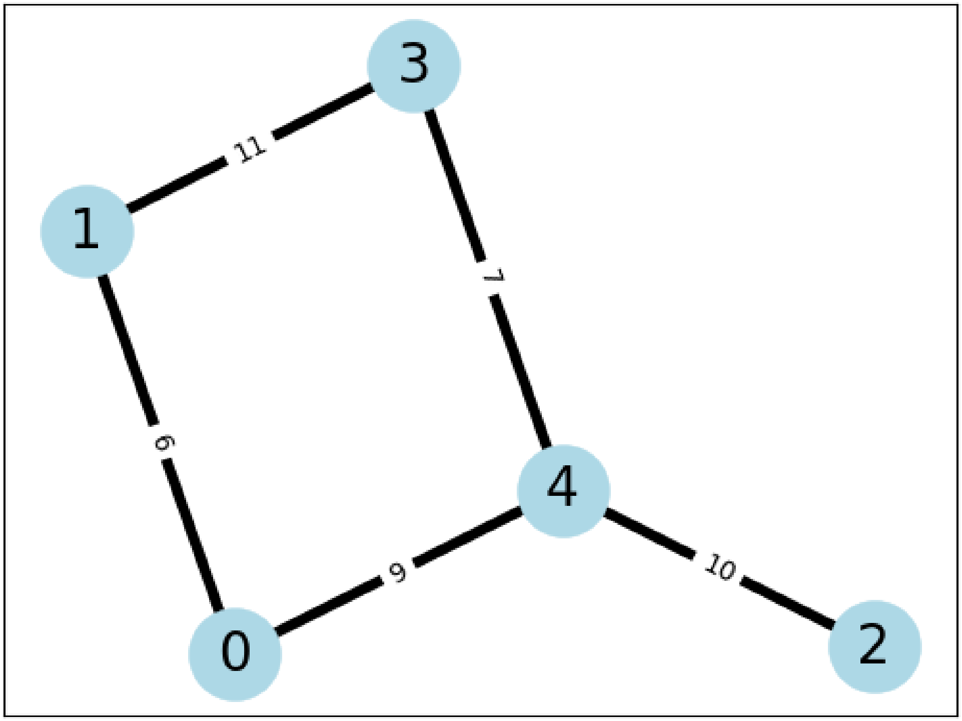
```
CityMap.get_city_map_statistics(r2_city_map)
```

```
node degree and node clustering
0 2 0
1 2 0
4 3 0
3 2 0
2 1 0

the adjacency list
```

```
0 1 4
1 3
4 2 3
3
2
```

[38]:
```
CityMap.visualize_city_map(r2_city_map)
```

```
{(0, 1): 6, (0, 4): 9, (1, 3): 11, (4, 2): 10, (4, 3): 7}
```



1 Generate Static Trips

[39]:
```
r2_trips = generate_static_trips()
len(r2_trips)
```

[39]:
```
20
```

2 R2

The benefit values of constructing the following new roads:

(0,2), (0,3), (1,2), (1,4), (2,3)

Use a k value (budget) of 2, which two of the above roads would you recommend for construction? Remember that once the first road is

constructed, benefits that you initially computed for the other 4 will now change and these will need to be recomputed.

Generate Benefit Matrix k = 0 # Initial Benefit Matrix

```
[40]: def get_max_benefit_road_segment(max_benefit_matrix: DataFrame) -> Tuple[int,␣
      ↪int]:
          max_benefit = max_benefit_matrix.iloc[0].values
          source = int(max_benefit[0])
          destination = int(max_benefit[1])

          return source, destination
```

```
[41]: r2_benefit_matrix, n1, n2, n1_n2_truth_table_data = TrafficAnalyzer.
      ↪get_road_recommendations(r2_city_map, r2_trips,debug=True)
      r2_benefit_matrix
```

```
[41]:    source  destination  benefit
      3       2            3     38.6
      4       0            2     38.0
      0       1            2     30.0
      2       1            4     30.0
      1       0            3     12.8
```

```
[42]: ### n1 and n2 Truth Tables - show details of internal algorithmic calculations
      n1_n2_truth_table_data
```

```
[42]:     x  y  nx_neighbor  ny_neighbor nx_indirect_benefits  \
      0   1  2            4           -1     {(4, 1), (1, 4)}
      1   1  2            4           -1     {(4, 1), (1, 4)}
      2   1  2           -1            0                   {}
      3   1  2           -1            0                   {}
      4   1  2           -1            3                   {}
      5   1  2           -1            3                   {}
      6   1  2           -1            3                   {}
      7   1  2           -1            3                   {}
      8   1  2           -1           -1                   {}
      9   0  3           -1           -1                   {}
      10  1  4            2           -1     {(1, 2), (2, 1)}
      11  1  4            2           -1     {(1, 2), (2, 1)}
      12  1  4            3           -1     {(1, 2), (2, 1)}
      13  1  4            3           -1     {(1, 2), (2, 1)}
      14  2  3            1           -1     {(1, 2), (2, 1)}
      15  2  3            1           -1     {(1, 2), (2, 1)}
      16  2  3            4           -1     {(1, 2), (2, 1)}
```

```
17  2  3            4           -1     {(1, 2), (2, 1)}
18  0  2           -1            1                   {}
19  0  2           -1            1                   {}
20  0  2           -1            4                   {}
21  0  2           -1            4                   {}

                ny_indirect_benefits  indirect_x  indirect_y  \
0                                 {}           4           1
1                                 {}           1           4
2                   {(0, 2), (2, 0)}           0           2
3                   {(0, 2), (2, 0)}           2           0
4   {(2, 3), (0, 2), (2, 0), (3, 2)}           2           3
5   {(2, 3), (0, 2), (2, 0), (3, 2)}           0           2
6   {(2, 3), (0, 2), (2, 0), (3, 2)}           2           0
7   {(2, 3), (0, 2), (2, 0), (3, 2)}           3           2
8                                 {}          -1          -1
9                                 {}          -1          -1
10                                {}           1           2
11                                {}           2           1
12                                {}           1           2
13                                {}           2           1
14                                {}           1           2
15                                {}           2           1
16                                {}           1           2
17                                {}           2           1
18                  {(1, 2), (2, 1)}           1           2
19                  {(1, 2), (2, 1)}           2           1
20                  {(1, 2), (2, 1)}           1           2
21                  {(1, 2), (2, 1)}           2           1

   has_edge_indirect_x_y has_edge_nx_neighbor_indirect_y  \
0                      T                               F
1                      F                               F
2                      F                               F
3                      F                               F
4                      F                               F
5                      F                               F
6                      F                               F
7                      F                               F
8                      F                               F
9                      F                               F
10                     F                               F
11                     T                               F
12                     F                               F
13                     T                               T
14                     T                               F
15                     F                               F
```

```
16                       T                                T
17                       F                                F
18                       F                                F
19                       F                                F
20                       F                                F
21                       F                                F

   has_edge_indirect_x_x has_edge_ny_neighbor_indirect_y
0                       F                                F
1                       F                                F
2                       T                                F
3                       F                                F
4                       F                                F
5                       T                                F
6                       F                                F
7                       T                                F
8                       F                                F
9                       F                                F
10                      F                                F
11                      F                                F
12                      F                                F
13                      F                                F
14                      F                                F
15                      F                                F
16                      F                                F
17                      F                                F
18                      T                                F
19                      F                                F
20                      T                                F
21                      F                                T
```

[43]:

```
##### k = 1 # Recommended road to build first is the road segment (2,3)
r2_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r2_benefit_matrix)
r2_max_benefit_matrix
```

[43]:

```
   source  destination  benefit
3       2            3     38.6
```

[44]:

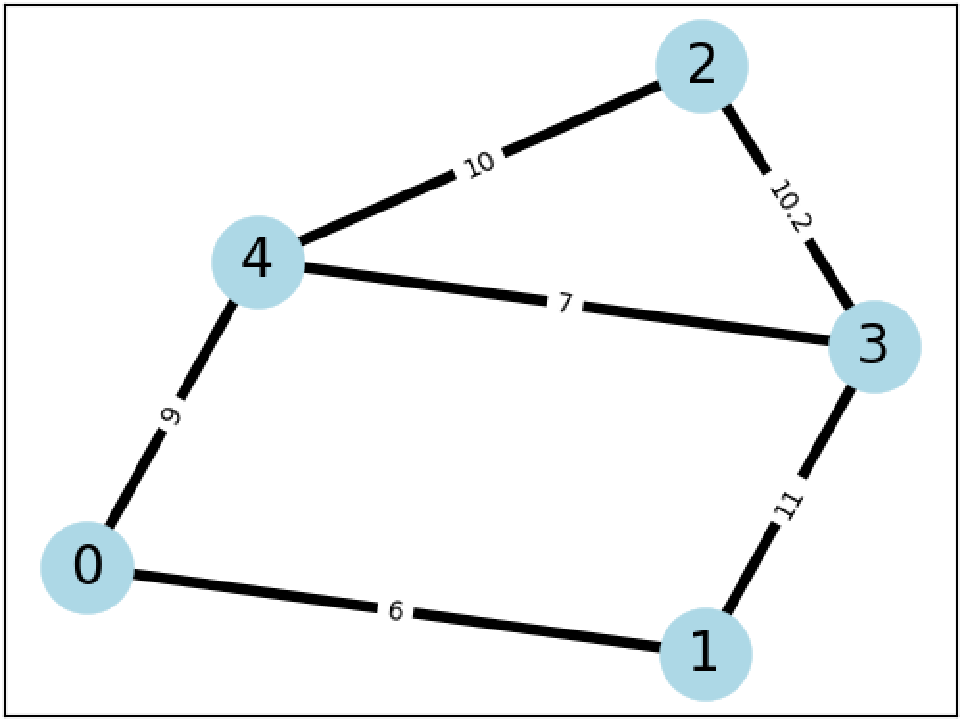
```
source, destination = get_max_benefit_road_segment(r2_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(2, 3)
```

[45]:

```
CityMap.add_road_segment(r2_city_map, source, destination)
CityMap.visualize_city_map(r2_city_map)
```

```
{(0, 1): 6, (0, 4): 9, (1, 3): 11, (4, 2): 10, (4, 3): 7, (3, 2): 10.2}
```



[46]:
```
r2_benefit_matrix, n1, n2, n1_n2_truth_table_data = TrafficAnalyzer.
 ↪get_road_recommendations(r2_city_map, r2_trips,debug=True)
r2_benefit_matrix
```

[46]:
```
   source  destination  benefit
1       0            2     26.6
0       1            2     26.0
3       1            4     18.6
2       0            3     12.8
```

[47]:
```
### n1 and n2 Truth Tables - show details of internal algorithmic calculations
n1_n2_truth_table_data
```

[47]:
```
   x  y  nx_neighbor  ny_neighbor nx_indirect_benefits ny_indirect_benefits  \
0  1  2            4           -1     {(4, 1), (1, 4)}                   {}
1  1  2            4           -1     {(4, 1), (1, 4)}                   {}
2  1  2           -1            0                   {}     {(0, 2), (2, 0)}
3  1  2           -1            0                   {}     {(0, 2), (2, 0)}
4  1  2           -1            3                   {}     {(0, 2), (2, 0)}
5  1  2           -1            3                   {}     {(0, 2), (2, 0)}
6  0  2            3           -1     {(0, 3), (3, 0)}                   {}
7  0  2            3           -1     {(0, 3), (3, 0)}                   {}
8  0  2            4           -1     {(0, 3), (3, 0)}                   {}
```

```
9   0  2            4           -1     {(0, 3), (3, 0)}                    {}
10  0  2           -1            1                   {}      {(1, 2), (2, 1)}
11  0  2           -1            1                   {}      {(1, 2), (2, 1)}
12  0  2           -1            4                   {}      {(1, 2), (2, 1)}
13  0  2           -1            4                   {}      {(1, 2), (2, 1)}
14  0  3            2           -1     {(0, 2), (2, 0)}                    {}
15  0  3            2           -1     {(0, 2), (2, 0)}                    {}
16  0  3            4           -1     {(0, 2), (2, 0)}                    {}
17  0  3            4           -1     {(0, 2), (2, 0)}                    {}
18  0  3           -1           -1                   {}                    {}
19  1  4            2           -1     {(1, 2), (2, 1)}                    {}
20  1  4            2           -1     {(1, 2), (2, 1)}                    {}
21  1  4            3           -1     {(1, 2), (2, 1)}                    {}
22  1  4            3           -1     {(1, 2), (2, 1)}                    {}

    indirect_x  indirect_y has_edge_indirect_x_y  \
0            4           1                     T
1            1           4                     F
2            0           2                     F
3            2           0                     F
4            0           2                     F
5            2           0                     F
6            0           3                     F
7            3           0                     T
8            0           3                     F
9            3           0                     T
10           1           2                     F
11           2           1                     F
12           1           2                     F
13           2           1                     F
14           0           2                     F
15           2           0                     T
16           0           2                     F
17           2           0                     T
18          -1          -1                     F
19           1           2                     F
20           2           1                     T
21           1           2                     F
22           2           1                     T

   has_edge_nx_neighbor_indirect_y has_edge_indirect_x_x  \
0                                F                     F
1                                F                     F
2                                F                     T
3                                F                     F
4                                F                     T
5                                F                     F
```

```
6                                       F                          F
7                                       F                          F
8                                       T                          F
9                                       T                          F
10                                      F                          T
11                                      F                          F
12                                      F                          T
13                                      F                          F
14                                      F                          F
15                                      F                          F
16                                      T                          F
17                                      T                          F
18                                      F                          F
19                                      F                          F
20                                      F                          F
21                                      T                          F
22                                      T                          F

   has_edge_ny_neighbor_indirect_y
0                                 F
1                                 F
2                                 F
3                                 F
4                                 F
5                                 T
6                                 F
7                                 F
8                                 F
9                                 F
10                                F
11                                F
12                                F
13                                T
14                                F
15                                F
16                                F
17                                F
18                                F
19                                F
20                                F
21                                F
22                                F
```

[48]:
```
##### k = 2 # Next recommended road to be built is the road segment (0,2)
r2_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r2_benefit_matrix)
r2_max_benefit_matrix
```

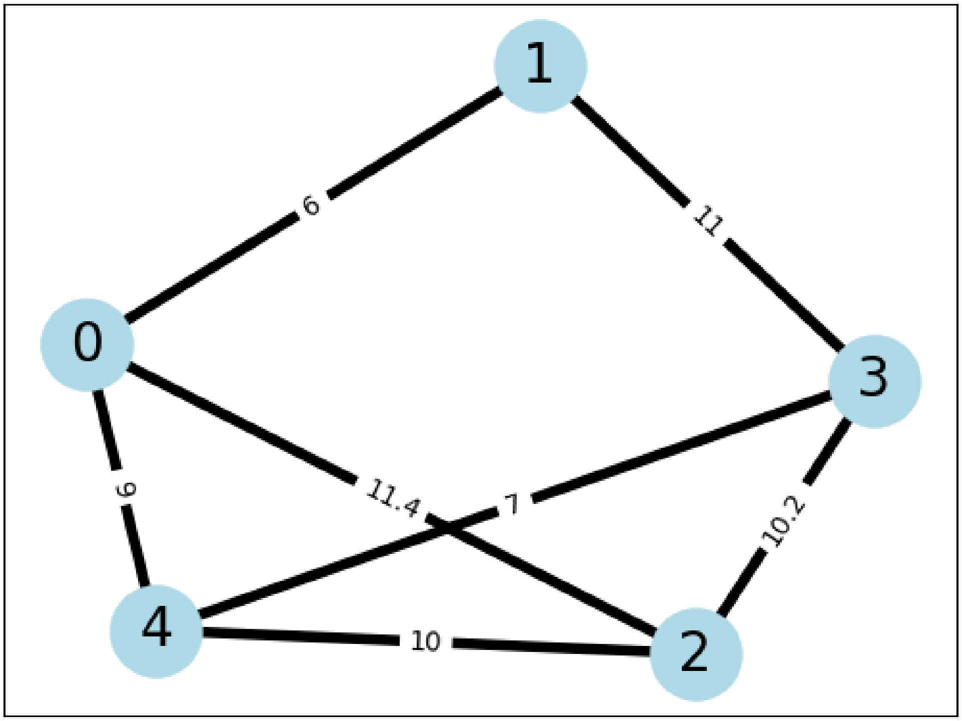
[48]:
```
   source  destination  benefit
1       0            2     26.6
```

[49]:
```
source, destination = get_max_benefit_road_segment(r2_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(0, 2)
```

[50]:
```
CityMap.add_road_segment(r2_city_map, source, destination)
CityMap.visualize_city_map(r2_city_map)
```

```
{(0, 1): 6, (0, 4): 9, (0, 2): 11.4, (1, 3): 11, (4, 2): 10, (4, 3): 7, (3, 2):
10.2}
```



3 R3

[56]:
```
r3_city_map = Simulator.generate_map()
r3_city_map
```

[56]:
```
<networkx.classes.graph.Graph at 0x128db6af0>
```

[57]:
```
r4_city_map = r3_city_map.copy()
r4_city_map
```

```
[57]: <networkx.classes.graph.Graph at 0x129492e20>
```

```
[58]: CityMap.get_city_map_statistics(r3_city_map)
```

```
node degree and node clustering
0 9 0.08333333333333333
1 9 0.1111111111111111
2 6 0.06666666666666667
3 6 0
4 8 0.14285714285714285
5 5 0
6 9 0.1388888888888889
7 5 0.1
8 11 0.16363636363636364
9 11 0.16363636363636364
10 8 0.03571428571428571
11 6 0.26666666666666666
12 12 0.13636363636363635
13 10 0.08888888888888889
14 16 0.11666666666666667
15 7 0.2857142857142857
16 9 0.05555555555555555
17 6 0.06666666666666667
18 11 0.07272727272727272
19 4 0.3333333333333333
20 6 0.06666666666666667
21 9 0.19444444444444445
22 5 0.1
23 8 0.21428571428571427
24 6 0.2
25 7 0.09523809523809523
26 6 0.2
27 9 0.2222222222222222
28 6 0.13333333333333333
29 11 0.16363636363636364
30 7 0.14285714285714285
31 13 0.16666666666666666
32 4 0.16666666666666666
33 7 0.047619047619047616
34 10 0.08888888888888889
35 5 0.1
36 1 0
37 4 0
38 3 0
39 7 0.14285714285714285
40 11 0.16363636363636364
41 7 0.09523809523809523
42 10 0.13333333333333333
```

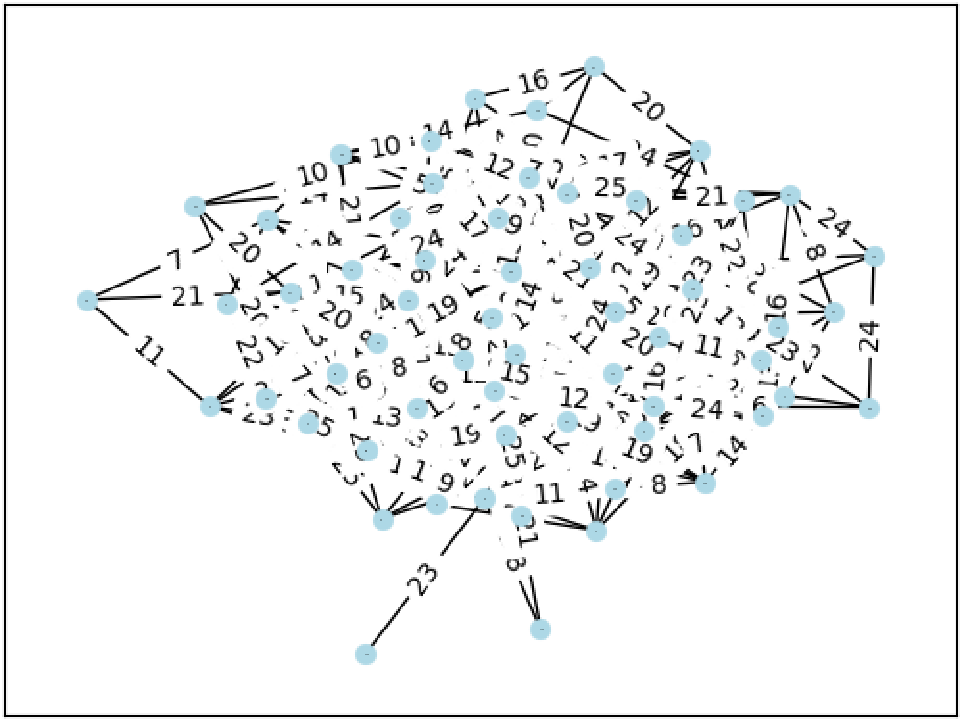
```
43 8 0.03571428571428571
44 8 0.14285714285714285
45 16 0.09166666666666666
46 2 0
47 10 0.13333333333333333
48 13 0.1794871794871795
49 10 0.1111111111111111
50 4 0
51 6 0.06666666666666667
52 8 0.17857142857142858
53 4 0.16666666666666666
54 9 0.08333333333333333
55 9 0.16666666666666666
56 11 0.2
57 6 0.13333333333333333
58 7 0.09523809523809523
59 9 0.16666666666666666

the adjacency list
0 20 3 35 59 14 27 47 32 5
1 12 52 50 58 39 15 13 8 22
2 11 3 54 8 48 20
3 10 38 13 31
4 22 14 55 6 44 5 45 15
5 18 31 54
6 25 45 31 12 46 27 21 36
7 12 18 22 25 45
8 14 23 40 35 52 51 15 34 42
9 31 28 56 49 48 55 46 11 47 45 58
10 29 16 45 54 50 30 42
11 34 48 42 14
12 31 43 55 21 27 53 18 14 29
13 49 56 17 40 14 21 39 34
14 23 42 22 59 30 52 21 16 29 31
15 42 52 54 55
16 18 53 20 48 34 55 43
17 38 45 39 30 42
18 58 52 21 50 47 28 59
19 57 49 53 45
20 24 51 49
21 30 54 35 28
22 57
23 43 31 37 29 40 54
24 29 43 48 49 26
25 48 33 58 59 42
26 45 59 39 33 29
27 49 34 31 41 40 56
28 33 40 30
```

```
29 51 45 34 47 44
30 44 59
31 39 34 40 56 47
32 47 44 41
33 49 39 51 34
34 47 44
35 49 42
36
37 56 50 51
38 47
39 41
40 54 45 56 58 48
41 45 56 52 43
42 48 43
43 47 45
44 56 57 45
45 51 49 57
46
47 59
48 53 49 59 55 56
49
50
51
52 54 57
53
54 58
55 57 58 56
56 59
57
58
59
```

[59]:
```
CityMap.visualize_city_map(r3_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
```

```
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55): 13, (16, 18):
15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16, 55): 16, (16,
43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17, 42): 11, (18,
58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13, (18, 28): 21,
(18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45): 9, (20, 24):
20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21, 35): 9, (21,
28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14, (23, 29): 5,
(23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23, (24, 49):
23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7, (25, 42):
12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29): 10, (27,
49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56): 8, (28,
33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34): 22,
(29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31, 34):
8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14, (32,
41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47): 15,
(34, 44): 16, (35, 49): 21, (35, 42): 25, (37, 56): 25, (37, 50): 16, (37, 51):
20, (38, 47): 21, (39, 41): 10, (40, 54): 8, (40, 45): 17, (40, 56): 5, (40,
58): 19, (40, 48): 7, (41, 45): 13, (41, 56): 7, (41, 52): 7, (41, 43): 12, (42,
48): 12, (42, 43): 14, (43, 47): 24, (43, 45): 20, (44, 56): 17, (44, 57): 25,
(44, 45): 9, (45, 51): 12, (45, 49): 25, (45, 57): 16, (47, 59): 20, (48, 53):
6, (48, 49): 16, (48, 59): 15, (48, 55): 24, (48, 56): 11, (52, 54): 24, (52,
57): 21, (54, 58): 19, (55, 57): 16, (55, 58): 14, (55, 56): 20, (56, 59): 19}
```



[60]:

```python
def get_traffic_times() -> Tuple[datetime, datetime]:
    # (8 AM - 6 PM) # 10 hour time span
    start_time = datetime.strptime('08:00', '%H:%M').time()
    end_time = datetime.strptime('18:00', '%H:%M').time()

    start_date = datetime.now() - timedelta(days=30)
    random_start_datetime = datetime.combine(start_date.date(), start_time)

    random_end_datetime = random_start_datetime + timedelta(hours=10)

    return random_start_datetime, random_end_datetime

traffic_start_datetime, traffic_end_datetime = get_traffic_times()

print("Traffic start datetime:", traffic_start_datetime)
print("Traffic end datetime", traffic_end_datetime)
```

```
Traffic start datetime: 2024-01-17 08:00:00
Traffic end datetime 2024-01-17 18:00:00
```

[63]:

```python
r3_r4_trips = Simulator.generate_trips(r3_city_map,
                                       traffic_start_datetime,
                                       traffic_end_datetime,
                                       TimeDeltaDiff.SECONDS)
```

```
r3_r4_number_of_trips = 0

for trip in r3_r4_trips:
    r3_r4_number_of_trips += trip.numer_of_trips

r3_r4_number_of_trips
```

```
[63]: 36000
```

```
[67]: r3_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r3_city_map,␣
        ↪r3_r4_trips)
      r3_benefit_matrix
```

```
[67]:       source  destination  benefit
      1003      36           45   2525.8
      25        14           37   1906.8
      278       31           36   1880.2
      527       38           45   1760.0
      570       22           31   1744.0
      …        …           …       …
      852       28           42    172.0
      164       10           53    168.4
      1446       5           28    158.4
      1355      26           32    149.6
      703        7           46     93.6

      [1535 rows x 3 columns]
```

```
[68]: ##### k = 1 # Recommended road to build first is the road segment
      r3_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r3_benefit_matrix)
      r3_max_benefit_matrix
```

```
[68]:       source  destination  benefit
      1003      36           45   2525.8
```

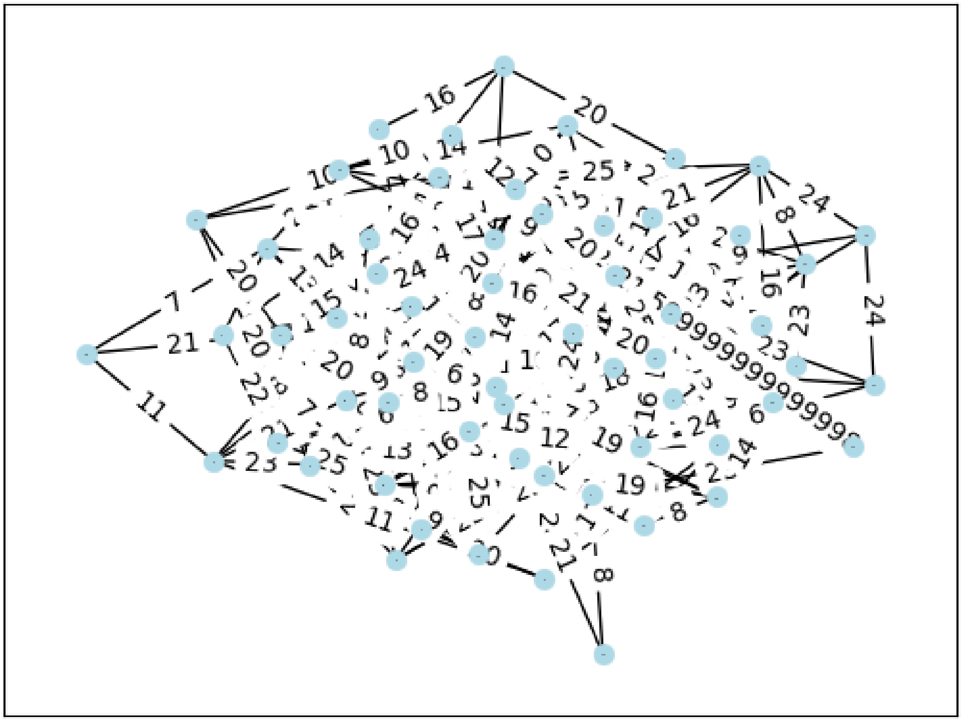
```
[69]: source, destination = get_max_benefit_road_segment(r3_max_benefit_matrix)
      print(f"({source}, {destination})")
```

```
(36, 45)
```

```
[70]: CityMap.add_road_segment(r3_city_map, source, destination)
```

```
[71]: CityMap.visualize_city_map(r3_city_map, location_size=60, location_font_size=1,␣
        ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55): 13, (16, 18):
15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16, 55): 16, (16,
43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17, 42): 11, (18,
58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13, (18, 28): 21,
(18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45): 9, (20, 24):
20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21, 35): 9, (21,
28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14, (23, 29): 5,
(23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23, (24, 49):
23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7, (25, 42):
12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29): 10, (27,
49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56): 8, (28,
33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34): 22,
(29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31, 34):
8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14, (32,
41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47): 15,
(34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (39, 41): 10, (40, 54): 8,
(40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13, (41, 56): 7,
(41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47): 24, (43, 45):
20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45, 49): 25, (45,
57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15, (48, 55): 24,
(48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57): 16, (55, 58):
14, (55, 56): 20, (56, 59): 19}
```



[72]:
```
r3_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r3_city_map,␣
  ↪r3_r4_trips)
r3_benefit_matrix
```

[72]:
```
      source  destination  benefit
527       38           45  1930.72
25        14           37  1906.80
1283      30           45  1831.44
570       22           31  1744.00
615       32           48  1616.60
…        …            …        …
852       28           42   172.00
164       10           53   168.40
1445       5           28   158.40
1354      26           32   149.60
703        7           46    93.60

[1534 rows x 3 columns]
```

[73]:
```
##### k = 2 # Recommended road to build first is the road segment
r3_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r3_benefit_matrix)
r3_max_benefit_matrix
```

[73]:

```
     source  destination  benefit
527      38           45  1930.72
```

[74]:

```
source, destination = get_max_benefit_road_segment(r3_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(38, 45)
```

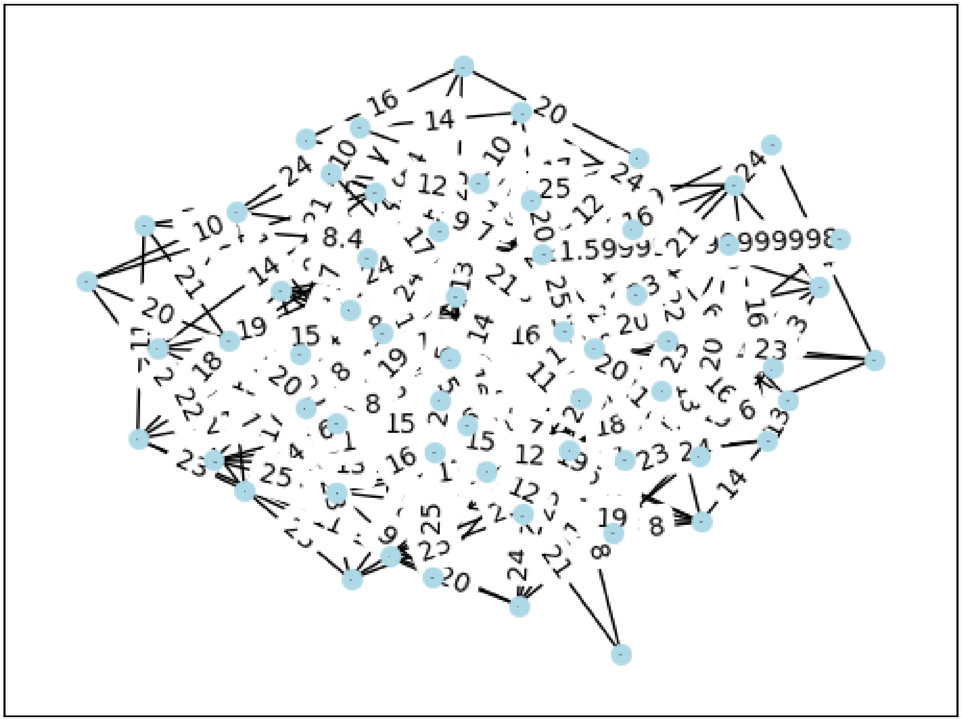
[75]:

```
CityMap.add_road_segment(r3_city_map, source, destination)
```

[76]:

```
CityMap.visualize_city_map(r3_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55): 13, (16, 18):
15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16, 55): 16, (16,
43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17, 42): 11, (18,
58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13, (18, 28): 21,
(18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45): 9, (20, 24):
20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21, 35): 9, (21,
28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14, (23, 29): 5,
(23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23, (24, 49):
23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7, (25, 42):
12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29): 10, (27,
49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56): 8, (28,
33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34): 22,
(29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31, 34):
8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14, (32,
41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47): 15,
(34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (38, 45): 8.4, (39, 41): 10,
(40, 54): 8, (40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13,
(41, 56): 7, (41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47):
24, (43, 45): 20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45,
49): 25, (45, 57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15,
(48, 55): 24, (48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57):
16, (55, 58): 14, (55, 56): 20, (56, 59): 19}
```



[77]:

```
r3_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r3_city_map,
  ↪r3_r4_trips)
r3_benefit_matrix
```

[77]:

```
      source  destination  benefit
25        14           37  1906.80
1282      30           45  1831.44
953        3           45  1769.04
318       45           46  1760.60
944       22           45  1643.52
…        …            …        …
164       10           53   168.40
1193       2           49   162.00
1444       5           28   158.40
1353      26           32   149.60
702        7           46    93.60
```

```
[1533 rows x 3 columns]
```

[78]:

```
##### k = 3 # Recommended road to build last is the road segment (14,18)
r3_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r3_benefit_matrix)
r3_max_benefit_matrix
```

[78]:

```
    source  destination  benefit
25      14           37   1906.8
```

[79]:

```
source, destination = get_max_benefit_road_segment(r3_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(14, 37)
```

[80]:

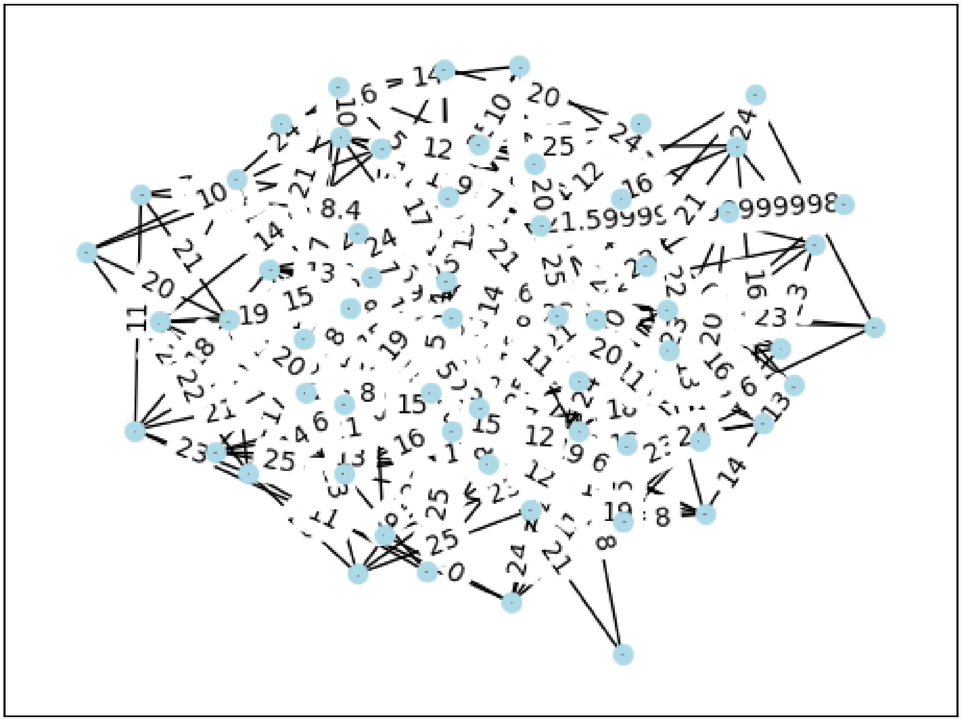
```
CityMap.add_road_segment(r3_city_map, source, destination)
```

[81]:

```
CityMap.visualize_city_map(r3_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (14, 37): 12.0, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55):
13, (16, 18): 15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16,
55): 16, (16, 43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17,
42): 11, (18, 58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13,
(18, 28): 21, (18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45):
9, (20, 24): 20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21,
35): 9, (21, 28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14,
(23, 29): 5, (23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23,
(24, 49): 23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7,
(25, 42): 12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29):
```

```
10, (27, 49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56):
8, (28, 33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34):
22, (29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31,
34): 8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14,
(32, 41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47):
15, (34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (38, 45): 8.4, (39, 41): 10,
(40, 54): 8, (40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13,
(41, 56): 7, (41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47):
24, (43, 45): 20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45,
49): 25, (45, 57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15,
(48, 55): 24, (48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57):
16, (55, 58): 14, (55, 56): 20, (56, 59): 19}
```



4 R4

```
[82]: r4_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r4_city_map,␣
        ↪r3_r4_trips, shrinkage_factor=0.8)
      r4_benefit_matrix
```

```
[82]:       source  destination  benefit
      1003      36           45   1139.6
      278       31           36    919.0
```

```
25         14            37    856.4
527        38            45    843.0
570        22            31    812.2
…       …          …       …
1446        5            28     67.2
544        34            52     62.4
375        56            58     57.6
1475        3            58     56.0
703         7            46     46.8

[1535 rows x 3 columns]
```

[83]:
```
##### k = 1 # Recommended road to build first is the road segment
r4_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r4_benefit_matrix)
r4_max_benefit_matrix
```

[83]:
```
      source  destination  benefit
1003      36           45   1139.6
```

[84]:
```
source, destination = get_max_benefit_road_segment(r4_max_benefit_matrix)
print(f"({source}, {destination})")
```

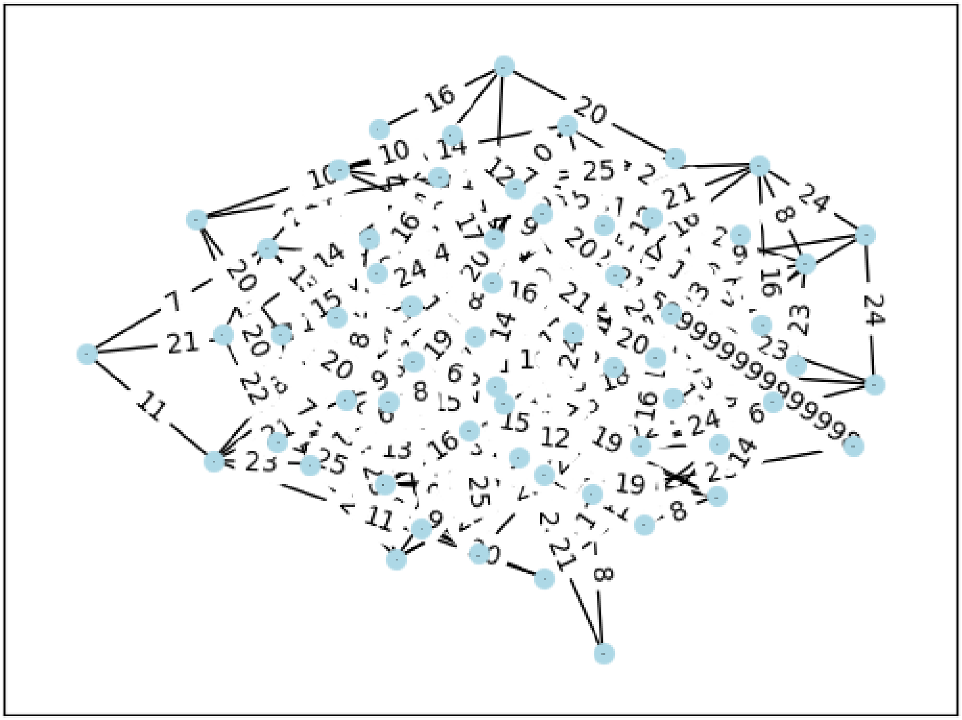
```
(36, 45)
```

[85]:
```
CityMap.add_road_segment(r4_city_map, source, destination)
```

[86]:
```
CityMap.visualize_city_map(r4_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55): 13, (16, 18):
```

```
15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16, 55): 16, (16,
43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17, 42): 11, (18,
58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13, (18, 28): 21,
(18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45): 9, (20, 24):
20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21, 35): 9, (21,
28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14, (23, 29): 5,
(23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23, (24, 49):
23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7, (25, 42):
12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29): 10, (27,
49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56): 8, (28,
33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34): 22,
(29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31, 34):
8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14, (32,
41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47): 15,
(34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (39, 41): 10, (40, 54): 8,
(40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13, (41, 56): 7,
(41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47): 24, (43, 45):
20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45, 49): 25, (45,
57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15, (48, 55): 24,
(48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57): 16, (55, 58):
14, (55, 56): 20, (56, 59): 19}
```



[87]:
```
r4_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r4_city_map,␣
 ↪r3_r4_trips, shrinkage_factor=0.8)
r4_benefit_matrix
```

[87]:
```
      source  destination  benefit
527       38           45   946.36
25        14           37   856.40
1283      30           45   841.72
570       22           31   812.20
954        3           45   694.04
…        …           …      …
1445       5           28    67.20
544       34           52    62.40
375       56           58    57.60
1474       3           58    56.00
703        7           46    46.80

[1534 rows x 3 columns]
```

[88]:
```
##### k = 2 # Recommended road to build first is the road segment
r4_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r4_benefit_matrix)
r4_max_benefit_matrix
```

[88]:
```
     source  destination  benefit
527      38           45   946.36
```

[89]:
```
source, destination = get_max_benefit_road_segment(r4_max_benefit_matrix)
print(f"({source}, {destination})")
```

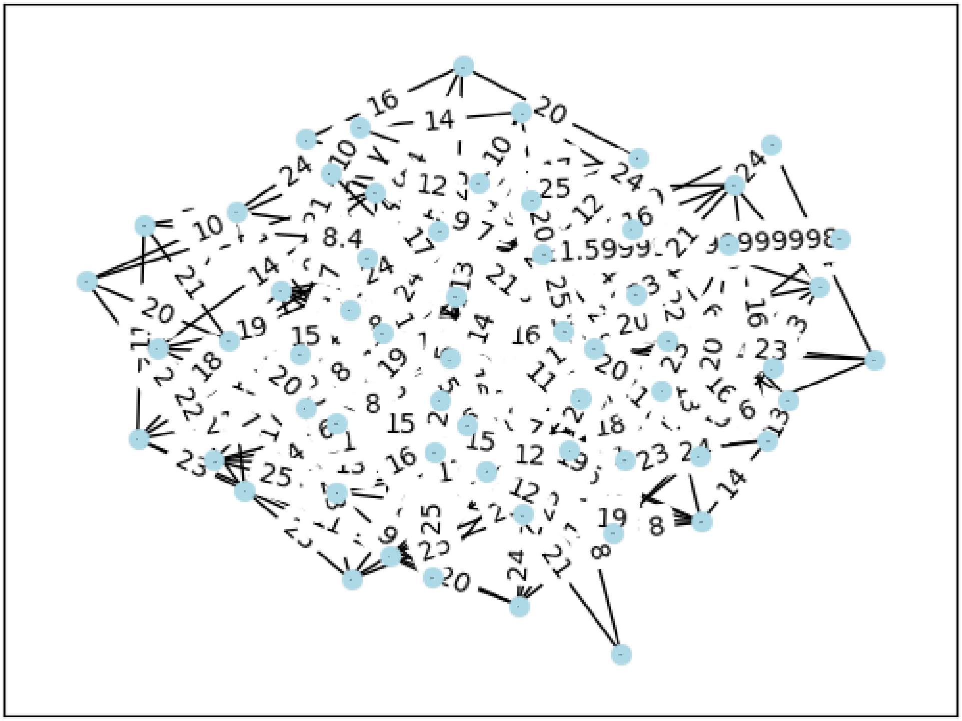
```
(38, 45)
```

[90]:
```
CityMap.add_road_segment(r4_city_map, source, destination)
```

[91]:
```
CityMap.visualize_city_map(r4_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
```

```
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55): 13, (16, 18):
15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16, 55): 16, (16,
43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17, 42): 11, (18,
58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13, (18, 28): 21,
(18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45): 9, (20, 24):
20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21, 35): 9, (21,
28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14, (23, 29): 5,
(23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23, (24, 49):
23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7, (25, 42):
12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29): 10, (27,
49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56): 8, (28,
33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34): 22,
(29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31, 34):
8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14, (32,
41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47): 15,
(34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (38, 45): 8.4, (39, 41): 10,
(40, 54): 8, (40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13,
(41, 56): 7, (41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47):
24, (43, 45): 20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45,
49): 25, (45, 57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15,
(48, 55): 24, (48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57):
16, (55, 58): 14, (55, 56): 20, (56, 59): 19}
```



```
[92]: r4_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r4_city_map,␣
        ↪r3_r4_trips, shrinkage_factor=0.8)
      r4_benefit_matrix
```

```
[92]:       source  destination  benefit
      25        14           37   856.40
      1282      30           45   841.72
      318       45           46   789.80
      953        3           45   772.04
      569       22           31   748.20
      …         …            …       …
      1444       5           28    67.20
      543       34           52    62.40
      375       56           58    57.60
      1473       3           58    56.00
      702        7           46    46.80

      [1533 rows x 3 columns]
```

```
[93]: ##### k = 3 # Recommended road to build last is the road segment
      r4_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r4_benefit_matrix)
      r4_max_benefit_matrix
```

[93]:

```
    source  destination  benefit
25      14           37    856.4
```

[94]:

```
source, destination = get_max_benefit_road_segment(r4_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(14, 37)
```

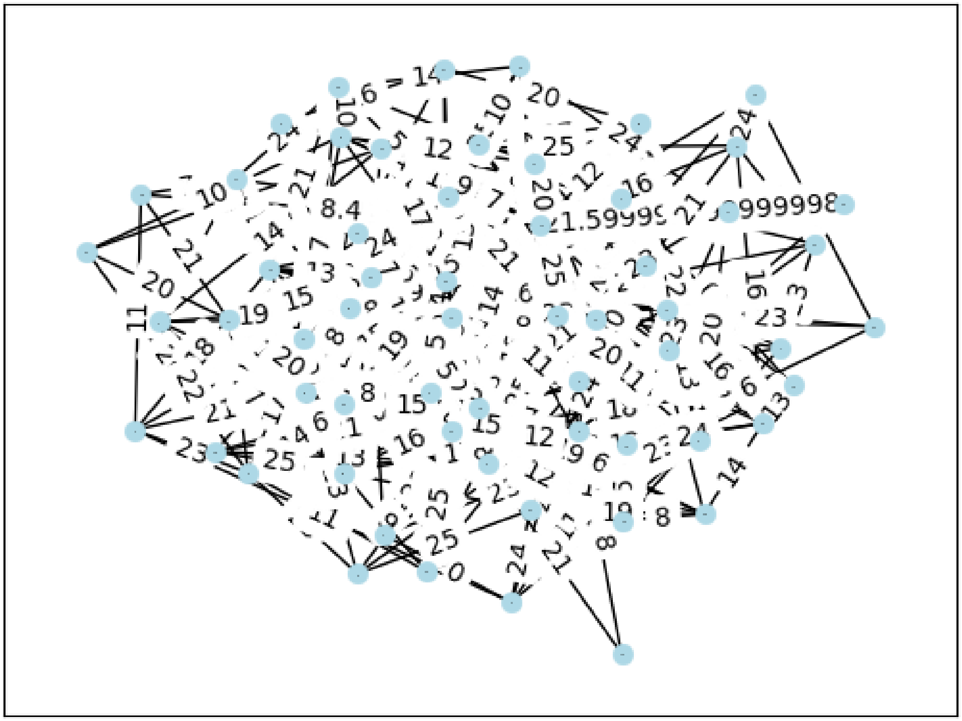
[95]:

```
CityMap.add_road_segment(r4_city_map, source, destination)
```

[96]:

```
CityMap.visualize_city_map(r4_city_map, location_size=60, location_font_size=1,␣
  ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 23, (0, 35): 5, (0, 59): 14, (0, 14): 24, (0, 27): 17, (0,
47): 12, (0, 32): 20, (0, 5): 25, (1, 12): 11, (1, 52): 23, (1, 50): 7, (1, 58):
7, (1, 39): 9, (1, 15): 6, (1, 13): 23, (1, 8): 19, (1, 22): 23, (2, 11): 20,
(2, 3): 11, (2, 54): 24, (2, 8): 23, (2, 48): 11, (2, 20): 18, (3, 10): 12, (3,
38): 11, (3, 13): 18, (3, 31): 21, (4, 22): 23, (4, 14): 18, (4, 55): 19, (4,
6): 5, (4, 44): 15, (4, 5): 15, (4, 45): 20, (4, 15): 6, (5, 18): 7, (5, 31):
23, (5, 54): 25, (6, 25): 11, (6, 45): 13, (6, 31): 13, (6, 12): 22, (6, 46): 8,
(6, 27): 12, (6, 21): 17, (6, 36): 23, (7, 12): 24, (7, 18): 22, (7, 22): 23,
(7, 25): 7, (7, 45): 19, (8, 14): 19, (8, 23): 9, (8, 40): 22, (8, 35): 13, (8,
52): 20, (8, 51): 15, (8, 15): 8, (8, 34): 8, (8, 42): 10, (9, 31): 15, (9, 28):
13, (9, 56): 16, (9, 49): 25, (9, 48): 5, (9, 55): 12, (9, 46): 21, (9, 11): 21,
(9, 47): 11, (9, 45): 18, (9, 58): 14, (10, 29): 11, (10, 16): 14, (10, 45): 15,
(10, 54): 11, (10, 50): 25, (10, 30): 19, (10, 42): 23, (11, 34): 18, (11, 48):
23, (11, 42): 21, (11, 14): 25, (12, 31): 11, (12, 43): 6, (12, 55): 11, (12,
21): 16, (12, 27): 16, (12, 53): 24, (12, 18): 16, (12, 14): 8, (12, 29): 23,
(13, 49): 9, (13, 56): 23, (13, 17): 13, (13, 40): 19, (13, 14): 23, (13, 21):
10, (13, 39): 21, (13, 34): 13, (14, 23): 6, (14, 42): 20, (14, 22): 20, (14,
59): 18, (14, 30): 17, (14, 52): 10, (14, 21): 8, (14, 16): 23, (14, 29): 13,
(14, 31): 9, (14, 37): 12.0, (15, 42): 18, (15, 52): 10, (15, 54): 23, (15, 55):
13, (16, 18): 15, (16, 53): 23, (16, 20): 13, (16, 48): 11, (16, 34): 16, (16,
55): 16, (16, 43): 24, (17, 38): 7, (17, 45): 7, (17, 39): 24, (17, 30): 5, (17,
42): 11, (18, 58): 16, (18, 52): 21, (18, 21): 16, (18, 50): 17, (18, 47): 13,
(18, 28): 21, (18, 59): 15, (19, 57): 24, (19, 49): 22, (19, 53): 24, (19, 45):
9, (20, 24): 20, (20, 51): 22, (20, 49): 11, (21, 30): 7, (21, 54): 19, (21,
35): 9, (21, 28): 25, (22, 57): 8, (23, 43): 23, (23, 31): 12, (23, 37): 14,
(23, 29): 5, (23, 40): 10, (23, 54): 5, (24, 29): 17, (24, 43): 8, (24, 48): 23,
(24, 49): 23, (24, 26): 24, (25, 48): 5, (25, 33): 22, (25, 58): 8, (25, 59): 7,
(25, 42): 12, (26, 45): 24, (26, 59): 20, (26, 39): 14, (26, 33): 25, (26, 29):
10, (27, 49): 16, (27, 34): 11, (27, 31): 9, (27, 41): 9, (27, 40): 6, (27, 56):
8, (28, 33): 24, (28, 40): 6, (28, 30): 22, (29, 51): 17, (29, 45): 9, (29, 34):
22, (29, 47): 24, (29, 44): 14, (30, 44): 14, (30, 59): 19, (31, 39): 7, (31,
34): 8, (31, 40): 8, (31, 56): 15, (31, 47): 13, (32, 47): 20, (32, 44): 14,
(32, 41): 10, (33, 49): 21, (33, 39): 5, (33, 51): 5, (33, 34): 24, (34, 47):
15, (34, 44): 16, (35, 49): 21, (35, 42): 25, (36, 45): 21.599999999999998, (37,
56): 25, (37, 50): 16, (37, 51): 20, (38, 47): 21, (38, 45): 8.4, (39, 41): 10,
(40, 54): 8, (40, 45): 17, (40, 56): 5, (40, 58): 19, (40, 48): 7, (41, 45): 13,
(41, 56): 7, (41, 52): 7, (41, 43): 12, (42, 48): 12, (42, 43): 14, (43, 47):
24, (43, 45): 20, (44, 56): 17, (44, 57): 25, (44, 45): 9, (45, 51): 12, (45,
49): 25, (45, 57): 16, (47, 59): 20, (48, 53): 6, (48, 49): 16, (48, 59): 15,
(48, 55): 24, (48, 56): 11, (52, 54): 24, (52, 57): 21, (54, 58): 19, (55, 57):
16, (55, 58): 14, (55, 56): 20, (56, 59): 19}
```



5 R5

[97]:
```
r5_city_map = Simulator.generate_map(connectedness=0.10)
r5_city_map
```

[97]:
```
<networkx.classes.graph.Graph at 0x12c479e50>
```

[98]:
```
CityMap.get_city_map_statistics(r5_city_map)
```

```
node degree and node clustering
0 9 0.08333333333333333
1 9 0.1111111111111111
2 6 0.06666666666666667
3 6 0
4 8 0.14285714285714285
5 5 0
6 9 0.1388888888888889
```

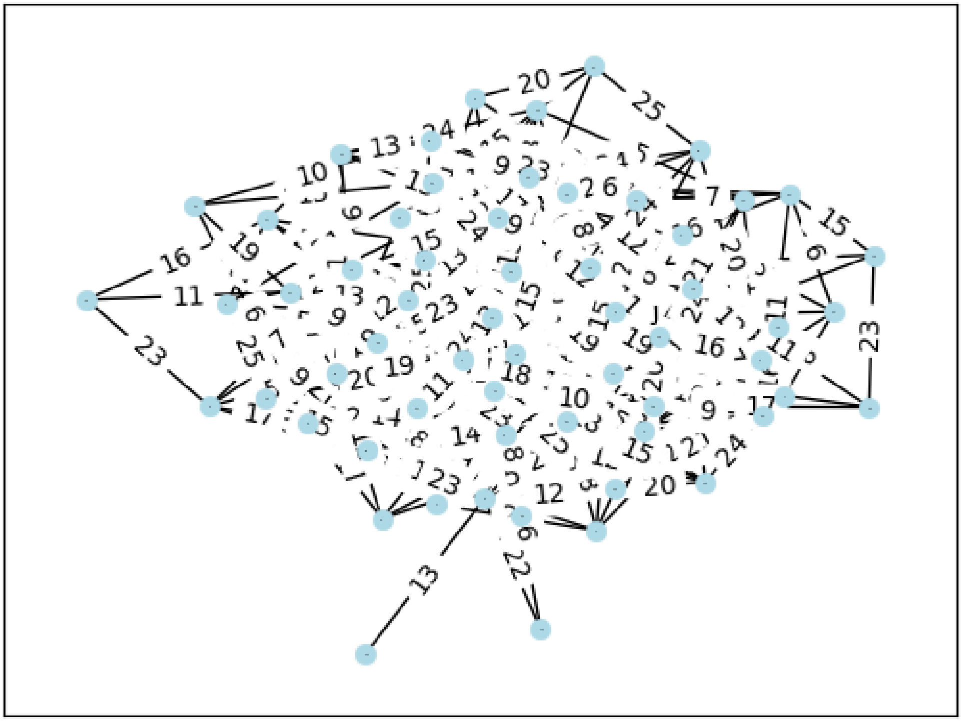
```
7 5 0.1
8 11 0.16363636363636364
9 11 0.16363636363636364
10 8 0.03571428571428571
11 6 0.26666666666666666
12 12 0.13636363636363635
13 10 0.08888888888888889
14 16 0.11666666666666667
15 7 0.2857142857142857
16 9 0.05555555555555555
17 6 0.06666666666666667
18 11 0.07272727272727272
19 4 0.3333333333333333
20 6 0.06666666666666667
21 9 0.19444444444444445
22 5 0.1
23 8 0.21428571428571427
24 6 0.2
25 7 0.09523809523809523
26 6 0.2
27 9 0.2222222222222222
28 6 0.13333333333333333
29 11 0.16363636363636364
30 7 0.14285714285714285
31 13 0.16666666666666666
32 4 0.16666666666666666
33 7 0.047619047619047616
34 10 0.08888888888888889
35 5 0.1
36 1 0
37 4 0
38 3 0
39 7 0.14285714285714285
40 11 0.16363636363636364
41 7 0.09523809523809523
42 10 0.13333333333333333
43 8 0.03571428571428571
44 8 0.14285714285714285
45 16 0.09166666666666666
46 2 0
47 10 0.13333333333333333
48 13 0.1794871794871795
49 10 0.1111111111111111
50 4 0
51 6 0.06666666666666667
52 8 0.17857142857142858
53 4 0.16666666666666666
54 9 0.08333333333333333
```

```
55 9 0.16666666666666666
56 11 0.2
57 6 0.13333333333333333
58 7 0.09523809523809523
59 9 0.16666666666666666

the adjacency list
0 20 3 35 59 14 27 47 32 5
1 12 52 50 58 39 15 13 8 22
2 11 3 54 8 48 20
3 10 38 13 31
4 22 14 55 6 44 5 45 15
5 18 31 54
6 25 45 31 12 46 27 21 36
7 12 18 22 25 45
8 14 23 40 35 52 51 15 34 42
9 31 28 56 49 48 55 46 11 47 45 58
10 29 16 45 54 50 30 42
11 34 48 42 14
12 31 43 55 21 27 53 18 14 29
13 49 56 17 40 14 21 39 34
14 23 42 22 59 30 52 21 16 29 31
15 42 52 54 55
16 18 53 20 48 34 55 43
17 38 45 39 30 42
18 58 52 21 50 47 28 59
19 57 49 53 45
20 24 51 49
21 30 54 35 28
22 57
23 43 31 37 29 40 54
24 29 43 48 49 26
25 48 33 58 59 42
26 45 59 39 33 29
27 49 34 31 41 40 56
28 33 40 30
29 51 45 34 47 44
30 44 59
31 39 34 40 56 47
32 47 44 41
33 49 39 51 34
34 47 44
35 49 42
36
37 56 50 51
38 47
39 41
40 54 45 56 58 48
```

```
41 45 56 52 43
42 48 43
43 47 45
44 56 57 45
45 51 49 57
46
47 59
48 53 49 59 55 56
49
50
51
52 54 57
53
54 58
55 57 58 56
56 59
57
58
59
```

```
[99]: CityMap.visualize_city_map(r5_city_map, location_size=60, location_font_size=1,␣
       ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 17, (0, 35): 20, (0, 59): 8, (0, 14): 13, (0, 27): 21, (0,
47): 8, (0, 32): 6, (0, 5): 17, (1, 12): 19, (1, 52): 5, (1, 50): 13, (1, 58):
18, (1, 39): 20, (1, 15): 15, (1, 13): 20, (1, 8): 6, (1, 22): 22, (2, 11): 24,
(2, 3): 7, (2, 54): 18, (2, 8): 25, (2, 48): 14, (2, 20): 20, (3, 10): 19, (3,
38): 23, (3, 13): 7, (3, 31): 5, (4, 22): 20, (4, 14): 20, (4, 55): 18, (4, 6):
20, (4, 44): 12, (4, 5): 5, (4, 45): 16, (4, 15): 11, (5, 18): 14, (5, 31): 12,
(5, 54): 19, (6, 25): 12, (6, 45): 9, (6, 31): 18, (6, 12): 20, (6, 46): 22, (6,
27): 10, (6, 21): 18, (6, 36): 13, (7, 12): 8, (7, 18): 24, (7, 22): 8, (7, 25):
21, (7, 45): 20, (8, 14): 8, (8, 23): 24, (8, 40): 7, (8, 35): 5, (8, 52): 21,
(8, 51): 20, (8, 15): 11, (8, 34): 5, (8, 42): 15, (9, 31): 12, (9, 28): 14, (9,
56): 5, (9, 49): 10, (9, 48): 19, (9, 55): 18, (9, 46): 6, (9, 11): 9, (9, 47):
14, (9, 45): 6, (9, 58): 12, (10, 29): 21, (10, 16): 21, (10, 45): 21, (10, 54):
19, (10, 50): 24, (10, 30): 23, (10, 42): 10, (11, 34): 14, (11, 48): 23, (11,
42): 12, (11, 14): 9, (12, 31): 25, (12, 43): 11, (12, 55): 18, (12, 21): 8,
(12, 27): 19, (12, 53): 7, (12, 18): 6, (12, 14): 13, (12, 29): 8, (13, 49): 21,
(13, 56): 8, (13, 17): 20, (13, 40): 8, (13, 14): 16, (13, 21): 12, (13, 39): 9,
(13, 34): 10, (14, 23): 21, (14, 42): 22, (14, 22): 14, (14, 59): 23, (14, 30):
7, (14, 52): 6, (14, 21): 15, (14, 16): 24, (14, 29): 15, (14, 31): 25, (15,
42): 13, (15, 52): 18, (15, 54): 8, (15, 55): 16, (16, 18): 17, (16, 53): 11,
(16, 20): 12, (16, 48): 17, (16, 34): 25, (16, 55): 17, (16, 43): 12, (17, 38):
16, (17, 45): 15, (17, 39): 20, (17, 30): 20, (17, 42): 25, (18, 58): 15, (18,
52): 23, (18, 21): 18, (18, 50): 11, (18, 47): 8, (18, 28): 22, (18, 59): 22,
(19, 57): 15, (19, 49): 15, (19, 53): 23, (19, 45): 14, (20, 24): 21, (20, 51):
20, (20, 49): 16, (21, 30): 9, (21, 54): 14, (21, 35): 23, (21, 28): 15, (22,
57): 6, (23, 43): 17, (23, 31): 7, (23, 37): 15, (23, 29): 25, (23, 40): 8, (23,
54): 16, (24, 29): 8, (24, 43): 6, (24, 48): 24, (24, 49): 21, (24, 26): 5, (25,
48): 6, (25, 33): 18, (25, 58): 20, (25, 59): 23, (25, 42): 25, (26, 45): 24,
(26, 59): 19, (26, 39): 24, (26, 33): 6, (26, 29): 8, (27, 49): 13, (27, 34):
11, (27, 31): 13, (27, 41): 25, (27, 40): 11, (27, 56): 24, (28, 33): 12, (28,
40): 20, (28, 30): 25, (29, 51): 24, (29, 45): 6, (29, 34): 8, (29, 47): 9, (29,
44): 11, (30, 44): 21, (30, 59): 5, (31, 39): 11, (31, 34): 9, (31, 40): 19,
(31, 56): 5, (31, 47): 7, (32, 47): 19, (32, 44): 15, (32, 41): 10, (33, 49):
12, (33, 39): 13, (33, 51): 23, (33, 34): 13, (34, 47): 13, (34, 44): 24, (35,
49): 14, (35, 42): 8, (37, 56): 24, (37, 50): 20, (37, 51): 25, (38, 47): 11,
(39, 41): 13, (40, 54): 11, (40, 45): 11, (40, 56): 13, (40, 58): 18, (40, 48):
18, (41, 45): 17, (41, 56): 18, (41, 52): 23, (41, 43): 9, (42, 48): 10, (42,
43): 15, (43, 47): 15, (43, 45): 8, (44, 56): 24, (44, 57): 6, (44, 45): 9, (45,
51): 24, (45, 49): 11, (45, 57): 16, (47, 59): 9, (48, 53): 17, (48, 49): 20,
(48, 59): 18, (48, 55): 9, (48, 56): 19, (52, 54): 15, (52, 57): 7, (54, 58):
15, (55, 57): 11, (55, 58): 24, (55, 56): 19, (56, 59): 23}
```



```
[101]: r5_trips = Simulator.generate_trips(r5_city_map,
                                  traffic_start_datetime,
                                  traffic_end_datetime,
                                  TimeDeltaDiff.SECONDS)
       r5_number_of_trips = 0

       for trip in r5_trips:
```

```
    r5_number_of_trips += trip.numer_of_trips

r5_number_of_trips
```

[101]:
```
36000
```

[103]:
```
r5_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r5_city_map,␣
 ↪r5_trips)
r5_benefit_matrix
```

[103]:
```
      source  destination  benefit
1003      36           45   2518.2
1125      45           54   1560.0
1521      45           58   1546.8
318       45           46   1528.4
1502      14           57   1520.2
…        …            …       …
1266      11           55    140.4
350       27           46    127.6
1520      34           49    124.8
1334      11           35    112.0
319       16           19    108.8

[1535 rows x 3 columns]
```

[104]:
```
##### k = 1 # Recommended road to build first is the road segment
r5_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r5_benefit_matrix)
r5_max_benefit_matrix
```

[104]:
```
      source  destination  benefit
1003      36           45   2518.2
```

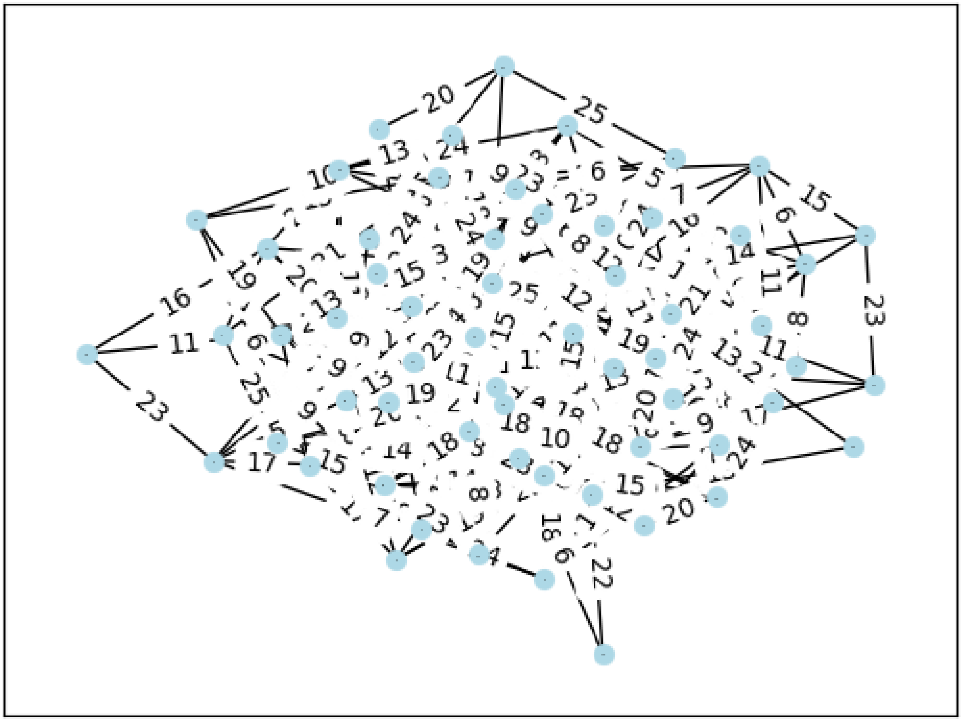
[105]:
```
source, destination = get_max_benefit_road_segment(r5_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(36, 45)
```

[106]:
```
CityMap.add_road_segment(r5_city_map, source, destination)
```

[107]:
```
CityMap.visualize_city_map(r5_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 17, (0, 35): 20, (0, 59): 8, (0, 14): 13, (0, 27): 21, (0,
47): 8, (0, 32): 6, (0, 5): 17, (1, 12): 19, (1, 52): 5, (1, 50): 13, (1, 58):
18, (1, 39): 20, (1, 15): 15, (1, 13): 20, (1, 8): 6, (1, 22): 22, (2, 11): 24,
(2, 3): 7, (2, 54): 18, (2, 8): 25, (2, 48): 14, (2, 20): 20, (3, 10): 19, (3,
38): 23, (3, 13): 7, (3, 31): 5, (4, 22): 20, (4, 14): 20, (4, 55): 18, (4, 6):
20, (4, 44): 12, (4, 5): 5, (4, 45): 16, (4, 15): 11, (5, 18): 14, (5, 31): 12,
(5, 54): 19, (6, 25): 12, (6, 45): 9, (6, 31): 18, (6, 12): 20, (6, 46): 22, (6,
27): 10, (6, 21): 18, (6, 36): 13, (7, 12): 8, (7, 18): 24, (7, 22): 8, (7, 25):
21, (7, 45): 20, (8, 14): 8, (8, 23): 24, (8, 40): 7, (8, 35): 5, (8, 52): 21,
(8, 51): 20, (8, 15): 11, (8, 34): 5, (8, 42): 15, (9, 31): 12, (9, 28): 14, (9,
56): 5, (9, 49): 10, (9, 48): 19, (9, 55): 18, (9, 46): 6, (9, 11): 9, (9, 47):
14, (9, 45): 6, (9, 58): 12, (10, 29): 21, (10, 16): 21, (10, 45): 21, (10, 54):
19, (10, 50): 24, (10, 30): 23, (10, 42): 10, (11, 34): 14, (11, 48): 23, (11,
42): 12, (11, 14): 9, (12, 31): 25, (12, 43): 11, (12, 55): 18, (12, 21): 8,
(12, 27): 19, (12, 53): 7, (12, 18): 6, (12, 14): 13, (12, 29): 8, (13, 49): 21,
(13, 56): 8, (13, 17): 20, (13, 40): 8, (13, 14): 16, (13, 21): 12, (13, 39): 9,
(13, 34): 10, (14, 23): 21, (14, 42): 22, (14, 22): 14, (14, 59): 23, (14, 30):
7, (14, 52): 6, (14, 21): 15, (14, 16): 24, (14, 29): 15, (14, 31): 25, (15,
42): 13, (15, 52): 18, (15, 54): 8, (15, 55): 16, (16, 18): 17, (16, 53): 11,
(16, 20): 12, (16, 48): 17, (16, 34): 25, (16, 55): 17, (16, 43): 12, (17, 38):
16, (17, 45): 15, (17, 39): 20, (17, 30): 20, (17, 42): 25, (18, 58): 15, (18,
52): 23, (18, 21): 18, (18, 50): 11, (18, 47): 8, (18, 28): 22, (18, 59): 22,
(19, 57): 15, (19, 49): 15, (19, 53): 23, (19, 45): 14, (20, 24): 21, (20, 51):
20, (20, 49): 16, (21, 30): 9, (21, 54): 14, (21, 35): 23, (21, 28): 15, (22,
57): 6, (23, 43): 17, (23, 31): 7, (23, 37): 15, (23, 29): 25, (23, 40): 8, (23,
54): 16, (24, 29): 8, (24, 43): 6, (24, 48): 24, (24, 49): 21, (24, 26): 5, (25,
48): 6, (25, 33): 18, (25, 58): 20, (25, 59): 23, (25, 42): 25, (26, 45): 24,
(26, 59): 19, (26, 39): 24, (26, 33): 6, (26, 29): 8, (27, 49): 13, (27, 34):
11, (27, 31): 13, (27, 41): 25, (27, 40): 11, (27, 56): 24, (28, 33): 12, (28,
40): 20, (28, 30): 25, (29, 51): 24, (29, 45): 6, (29, 34): 8, (29, 47): 9, (29,
44): 11, (30, 44): 21, (30, 59): 5, (31, 39): 11, (31, 34): 9, (31, 40): 19,
(31, 56): 5, (31, 47): 7, (32, 47): 19, (32, 44): 15, (32, 41): 10, (33, 49):
12, (33, 39): 13, (33, 51): 23, (33, 34): 13, (34, 47): 13, (34, 44): 24, (35,
49): 14, (35, 42): 8, (36, 45): 13.2, (37, 56): 24, (37, 50): 20, (37, 51): 25,
(38, 47): 11, (39, 41): 13, (40, 54): 11, (40, 45): 11, (40, 56): 13, (40, 58):
18, (40, 48): 18, (41, 45): 17, (41, 56): 18, (41, 52): 23, (41, 43): 9, (42,
48): 10, (42, 43): 15, (43, 47): 15, (43, 45): 8, (44, 56): 24, (44, 57): 6,
(44, 45): 9, (45, 51): 24, (45, 49): 11, (45, 57): 16, (47, 59): 9, (48, 53):
17, (48, 49): 20, (48, 59): 18, (48, 55): 9, (48, 56): 19, (52, 54): 15, (52,
57): 7, (54, 58): 15, (55, 57): 11, (55, 58): 24, (55, 56): 19, (56, 59): 23}
```



[108]:
```
r5_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r5_city_map,␣
 ↪r5_trips)
r5_benefit_matrix
```

[108]:
```
      source  destination  benefit
1520      45           58  1768.80
1124      45           54  1586.64
1297      11           45  1584.20
318       45           46  1577.68
1397      28           45  1569.20
…        …           …       …
350       27           46   127.60
1519      34           49   124.80
1333      11           35   112.00
1349      36           46   110.88
319       16           19   108.80

[1534 rows x 3 columns]
```

[109]:
```
##### k = 2 # Recommended road to build first is the road segment
r5_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r5_benefit_matrix)
r5_max_benefit_matrix
```

```
[109]:       source  destination  benefit
       1520      45           58   1768.8
```

```
[110]: source, destination = get_max_benefit_road_segment(r5_max_benefit_matrix)
       print(f"({source}, {destination})")
```

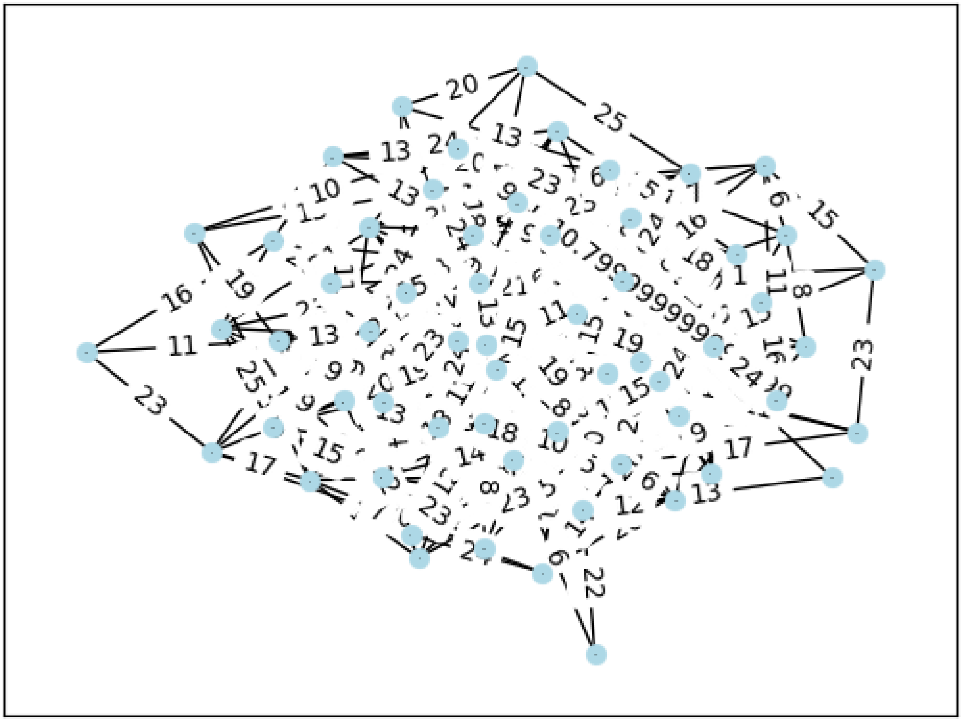
```
(45, 58)
```

```
[111]: CityMap.add_road_segment(r5_city_map, source, destination)
```

```
[112]: CityMap.visualize_city_map(r5_city_map, location_size=60, location_font_size=1,␣
         ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 17, (0, 35): 20, (0, 59): 8, (0, 14): 13, (0, 27): 21, (0,
47): 8, (0, 32): 6, (0, 5): 17, (1, 12): 19, (1, 52): 5, (1, 50): 13, (1, 58):
18, (1, 39): 20, (1, 15): 15, (1, 13): 20, (1, 8): 6, (1, 22): 22, (2, 11): 24,
(2, 3): 7, (2, 54): 18, (2, 8): 25, (2, 48): 14, (2, 20): 20, (3, 10): 19, (3,
38): 23, (3, 13): 7, (3, 31): 5, (4, 22): 20, (4, 14): 20, (4, 55): 18, (4, 6):
20, (4, 44): 12, (4, 5): 5, (4, 45): 16, (4, 15): 11, (5, 18): 14, (5, 31): 12,
(5, 54): 19, (6, 25): 12, (6, 45): 9, (6, 31): 18, (6, 12): 20, (6, 46): 22, (6,
27): 10, (6, 21): 18, (6, 36): 13, (7, 12): 8, (7, 18): 24, (7, 22): 8, (7, 25):
21, (7, 45): 20, (8, 14): 8, (8, 23): 24, (8, 40): 7, (8, 35): 5, (8, 52): 21,
(8, 51): 20, (8, 15): 11, (8, 34): 5, (8, 42): 15, (9, 31): 12, (9, 28): 14, (9,
56): 5, (9, 49): 10, (9, 48): 19, (9, 55): 18, (9, 46): 6, (9, 11): 9, (9, 47):
14, (9, 45): 6, (9, 58): 12, (10, 29): 21, (10, 16): 21, (10, 45): 21, (10, 54):
19, (10, 50): 24, (10, 30): 23, (10, 42): 10, (11, 34): 14, (11, 48): 23, (11,
42): 12, (11, 14): 9, (12, 31): 25, (12, 43): 11, (12, 55): 18, (12, 21): 8,
(12, 27): 19, (12, 53): 7, (12, 18): 6, (12, 14): 13, (12, 29): 8, (13, 49): 21,
(13, 56): 8, (13, 17): 20, (13, 40): 8, (13, 14): 16, (13, 21): 12, (13, 39): 9,
(13, 34): 10, (14, 23): 21, (14, 42): 22, (14, 22): 14, (14, 59): 23, (14, 30):
7, (14, 52): 6, (14, 21): 15, (14, 16): 24, (14, 29): 15, (14, 31): 25, (15,
42): 13, (15, 52): 18, (15, 54): 8, (15, 55): 16, (16, 18): 17, (16, 53): 11,
(16, 20): 12, (16, 48): 17, (16, 34): 25, (16, 55): 17, (16, 43): 12, (17, 38):
16, (17, 45): 15, (17, 39): 20, (17, 30): 20, (17, 42): 25, (18, 58): 15, (18,
52): 23, (18, 21): 18, (18, 50): 11, (18, 47): 8, (18, 28): 22, (18, 59): 22,
(19, 57): 15, (19, 49): 15, (19, 53): 23, (19, 45): 14, (20, 24): 21, (20, 51):
20, (20, 49): 16, (21, 30): 9, (21, 54): 14, (21, 35): 23, (21, 28): 15, (22,
57): 6, (23, 43): 17, (23, 31): 7, (23, 37): 15, (23, 29): 25, (23, 40): 8, (23,
54): 16, (24, 29): 8, (24, 43): 6, (24, 48): 24, (24, 49): 21, (24, 26): 5, (25,
48): 6, (25, 33): 18, (25, 58): 20, (25, 59): 23, (25, 42): 25, (26, 45): 24,
(26, 59): 19, (26, 39): 24, (26, 33): 6, (26, 29): 8, (27, 49): 13, (27, 34):
11, (27, 31): 13, (27, 41): 25, (27, 40): 11, (27, 56): 24, (28, 33): 12, (28,
40): 20, (28, 30): 25, (29, 51): 24, (29, 45): 6, (29, 34): 8, (29, 47): 9, (29,
44): 11, (30, 44): 21, (30, 59): 5, (31, 39): 11, (31, 34): 9, (31, 40): 19,
(31, 56): 5, (31, 47): 7, (32, 47): 19, (32, 44): 15, (32, 41): 10, (33, 49):
12, (33, 39): 13, (33, 51): 23, (33, 34): 13, (34, 47): 13, (34, 44): 24, (35,
49): 14, (35, 42): 8, (36, 45): 13.2, (37, 56): 24, (37, 50): 20, (37, 51): 25,
(38, 47): 11, (39, 41): 13, (40, 54): 11, (40, 45): 11, (40, 56): 13, (40, 58):
```

```
18, (40, 48): 18, (41, 45): 17, (41, 56): 18, (41, 52): 23, (41, 43): 9, (42,
48): 10, (42, 43): 15, (43, 47): 15, (43, 45): 8, (44, 56): 24, (44, 57): 6,
(44, 45): 9, (45, 51): 24, (45, 49): 11, (45, 57): 16, (45, 58):
10.799999999999999, (47, 59): 9, (48, 53): 17, (48, 49): 20, (48, 59): 18, (48,
55): 9, (48, 56): 19, (52, 54): 15, (52, 57): 7, (54, 58): 15, (55, 57): 11,
(55, 58): 24, (55, 56): 19, (56, 59): 23}
```



[113]:
```
r5_benefit_matrix = TrafficAnalyzer.get_road_recommendations(r5_city_map,␣
↪r5_trips)
r5_benefit_matrix
```

[113]:
```
      source  destination  benefit
374       37           45  1753.24
831        5           45  1627.80
304       23           45  1626.20
1124      45           54  1586.64
318       45           46  1577.68
…        …            …       …
350       27           46   127.60
1519      34           49   124.80
1333      11           35   112.00
1349      36           46   110.88
319       16           19   108.80
```

```
[1533 rows x 3 columns]
```

[114]:

```
##### k = 3 # Recommended road to build last is the road segment
r5_max_benefit_matrix = TrafficAnalyzer.get_max_road_benefit(r5_benefit_matrix)
r5_max_benefit_matrix
```

[114]:

```
     source  destination  benefit
374      37           45  1753.24
```

[115]:

```
source, destination = get_max_benefit_road_segment(r5_max_benefit_matrix)
print(f"({source}, {destination})")
```

```
(37, 45)
```

[116]:

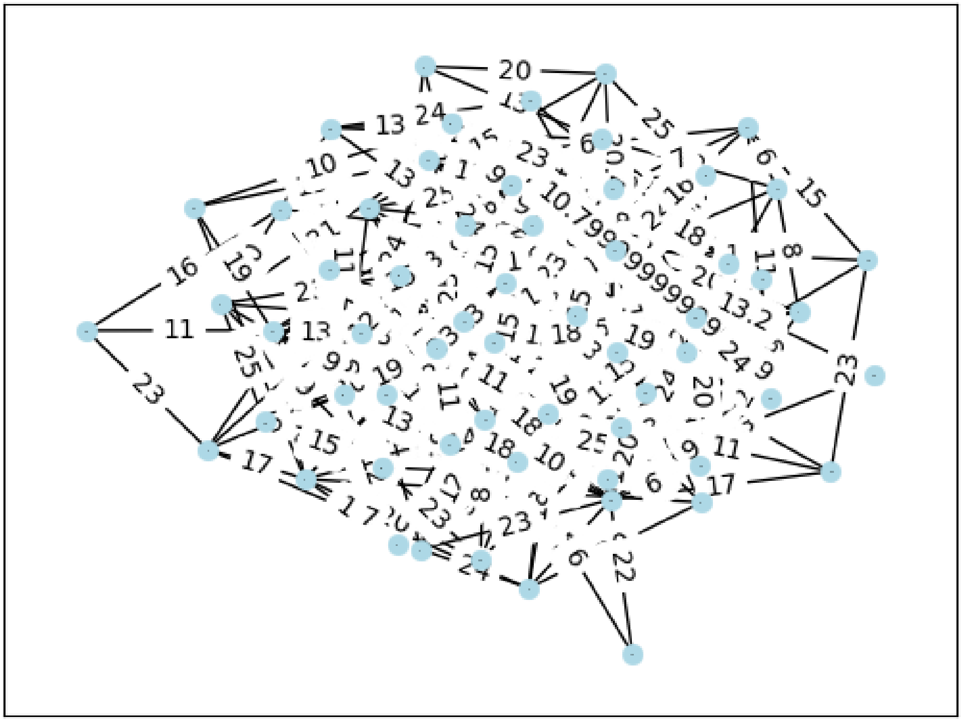
```
CityMap.add_road_segment(r5_city_map, source, destination)
```

[117]:

```
CityMap.visualize_city_map(r5_city_map, location_size=60, location_font_size=1,␣
 ↪road_widths=1)
```

```
{(0, 20): 8, (0, 3): 17, (0, 35): 20, (0, 59): 8, (0, 14): 13, (0, 27): 21, (0,
47): 8, (0, 32): 6, (0, 5): 17, (1, 12): 19, (1, 52): 5, (1, 50): 13, (1, 58):
18, (1, 39): 20, (1, 15): 15, (1, 13): 20, (1, 8): 6, (1, 22): 22, (2, 11): 24,
(2, 3): 7, (2, 54): 18, (2, 8): 25, (2, 48): 14, (2, 20): 20, (3, 10): 19, (3,
38): 23, (3, 13): 7, (3, 31): 5, (4, 22): 20, (4, 14): 20, (4, 55): 18, (4, 6):
20, (4, 44): 12, (4, 5): 5, (4, 45): 16, (4, 15): 11, (5, 18): 14, (5, 31): 12,
(5, 54): 19, (6, 25): 12, (6, 45): 9, (6, 31): 18, (6, 12): 20, (6, 46): 22, (6,
27): 10, (6, 21): 18, (6, 36): 13, (7, 12): 8, (7, 18): 24, (7, 22): 8, (7, 25):
21, (7, 45): 20, (8, 14): 8, (8, 23): 24, (8, 40): 7, (8, 35): 5, (8, 52): 21,
(8, 51): 20, (8, 15): 11, (8, 34): 5, (8, 42): 15, (9, 31): 12, (9, 28): 14, (9,
56): 5, (9, 49): 10, (9, 48): 19, (9, 55): 18, (9, 46): 6, (9, 11): 9, (9, 47):
14, (9, 45): 6, (9, 58): 12, (10, 29): 21, (10, 16): 21, (10, 45): 21, (10, 54):
19, (10, 50): 24, (10, 30): 23, (10, 42): 10, (11, 34): 14, (11, 48): 23, (11,
42): 12, (11, 14): 9, (12, 31): 25, (12, 43): 11, (12, 55): 18, (12, 21): 8,
(12, 27): 19, (12, 53): 7, (12, 18): 6, (12, 14): 13, (12, 29): 8, (13, 49): 21,
(13, 56): 8, (13, 17): 20, (13, 40): 8, (13, 14): 16, (13, 21): 12, (13, 39): 9,
(13, 34): 10, (14, 23): 21, (14, 42): 22, (14, 22): 14, (14, 59): 23, (14, 30):
7, (14, 52): 6, (14, 21): 15, (14, 16): 24, (14, 29): 15, (14, 31): 25, (15,
42): 13, (15, 52): 18, (15, 54): 8, (15, 55): 16, (16, 18): 17, (16, 53): 11,
(16, 20): 12, (16, 48): 17, (16, 34): 25, (16, 55): 17, (16, 43): 12, (17, 38):
16, (17, 45): 15, (17, 39): 20, (17, 30): 20, (17, 42): 25, (18, 58): 15, (18,
52): 23, (18, 21): 18, (18, 50): 11, (18, 47): 8, (18, 28): 22, (18, 59): 22,
(19, 57): 15, (19, 49): 15, (19, 53): 23, (19, 45): 14, (20, 24): 21, (20, 51):
20, (20, 49): 16, (21, 30): 9, (21, 54): 14, (21, 35): 23, (21, 28): 15, (22,
57): 6, (23, 43): 17, (23, 31): 7, (23, 37): 15, (23, 29): 25, (23, 40): 8, (23,
54): 16, (24, 29): 8, (24, 43): 6, (24, 48): 24, (24, 49): 21, (24, 26): 5, (25,
48): 6, (25, 33): 18, (25, 58): 20, (25, 59): 23, (25, 42): 25, (26, 45): 24,
(26, 59): 19, (26, 39): 24, (26, 33): 6, (26, 29): 8, (27, 49): 13, (27, 34):
```

```
11, (27, 31): 13, (27, 41): 25, (27, 40): 11, (27, 56): 24, (28, 33): 12, (28,
40): 20, (28, 30): 25, (29, 51): 24, (29, 45): 6, (29, 34): 8, (29, 47): 9, (29,
44): 11, (30, 44): 21, (30, 59): 5, (31, 39): 11, (31, 34): 9, (31, 40): 19,
(31, 56): 5, (31, 47): 7, (32, 47): 19, (32, 44): 15, (32, 41): 10, (33, 49):
12, (33, 39): 13, (33, 51): 23, (33, 34): 13, (34, 47): 13, (34, 44): 24, (35,
49): 14, (35, 42): 8, (36, 45): 13.2, (37, 56): 24, (37, 50): 20, (37, 51): 25,
(37, 45): 20.4, (38, 47): 11, (39, 41): 13, (40, 54): 11, (40, 45): 11, (40,
56): 13, (40, 58): 18, (40, 48): 18, (41, 45): 17, (41, 56): 18, (41, 52): 23,
(41, 43): 9, (42, 48): 10, (42, 43): 15, (43, 47): 15, (43, 45): 8, (44, 56):
24, (44, 57): 6, (44, 45): 9, (45, 51): 24, (45, 49): 11, (45, 57): 16, (45,
58): 10.799999999999999, (47, 59): 9, (48, 53): 17, (48, 49): 20, (48, 59): 18,
(48, 55): 9, (48, 56): 19, (52, 54): 15, (52, 57): 7, (54, 58): 15, (55, 57):
11, (55, 58): 24, (55, 56): 19, (56, 59): 23}
```


20
13
24
25
10
16
19
11
13
23
25
17
15
10.799999999999999
13.2
8
15
23
11
17
6
22
24
23
19
18