Assignment 2A - Hill Climbing

In this assignment, you will implement the necessary data structures and subroutines to run hill climbing on traveling salesman problems.

You are given four files

1. This Notebook - You will be running the cells in this file.
2. sa_utils.py. You should not change anything in this file.
3. tsp_utils.py. You will be writing most of the code in this file.
4. berlin52.tsp. A TSP file, downloaded from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

TODO

Enter your information below.

Name: HaoLiu
CWID: A20473685

In [1]:
```
import numpy as np

import pandas as pd

from matplotlib import pylab
import matplotlib.pyplot as plt
pylab.rcParams['figure.figsize'] = (10.0, 8.0)
from sa_utils import Node
from sa_utils import hill_climbing
```

In [2]:
```
from tsp_utils import City, TSPNode, read_cities, subsample_cities, create_initial_node
from tsp_utils import plot_cities, plot_path, compare_sols
```

Reading the file

TODO: Implement the `read_cities` function in `tsp_utils.py`. This function should read a given TSP file from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Check out the documentation file. Your function should support only the EUC_2D types of files. See `berlin52.tsp` as an example.

`read_cities` should accept a string (filename) and return a dictionary of the City objects, where the key is the City name and the objects are City objects with the correct coordinates.

In [3]:
```
# Run. It should show the dictionary.
all_cities = read_cities('berlin52.tsp')
TSPNode._cities = all_cities
all_cities
```

Out[3]:
```
{'1': City: 1 (565.00 575.00),
 '2': City: 2 (25.00 185.00),
 '3': City: 3 (345.00 750.00),
 '4': City: 4 (945.00 685.00),
 '5': City: 5 (845.00 655.00),
 '6': City: 6 (880.00 660.00),
 '7': City: 7 (25.00 230.00),
 '8': City: 8 (525.00 1000.00),
 '9': City: 9 (580.00 1175.00),
 '10': City: 10 (650.00 1130.00),
 '11': City: 11 (1605.00 620.00),
 '12': City: 12 (1220.00 580.00),
 '13': City: 13 (1465.00 200.00),
 '14': City: 14 (1530.00 5.00),
 '15': City: 15 (845.00 680.00),
 '16': City: 16 (725.00 370.00),
 '17': City: 17 (145.00 665.00),
 '18': City: 18 (415.00 635.00),
 '19': City: 19 (510.00 875.00),
 '20': City: 20 (560.00 365.00),
 '21': City: 21 (300.00 465.00),
 '22': City: 22 (520.00 585.00),
 '23': City: 23 (480.00 415.00),
 '24': City: 24 (835.00 625.00),
 '25': City: 25 (975.00 580.00),
 '26': City: 26 (1215.00 245.00),
 '27': City: 27 (1320.00 315.00),
 '28': City: 28 (1250.00 400.00),
 '29': City: 29 (660.00 180.00),
 '30': City: 30 (410.00 250.00),
 '31': City: 31 (420.00 555.00),
 '32': City: 32 (575.00 665.00),
 '33': City: 33 (1150.00 1160.00),
 '34': City: 34 (700.00 580.00),
 '35': City: 35 (685.00 595.00),
 '36': City: 36 (685.00 610.00),
 '37': City: 37 (770.00 610.00),
 '38': City: 38 (795.00 645.00),
 '39': City: 39 (720.00 635.00),
 '40': City: 40 (760.00 650.00),
 '41': City: 41 (475.00 960.00),
 '42': City: 42 (95.00 260.00),
 '43': City: 43 (875.00 920.00),
 '44': City: 44 (700.00 500.00),
 '45': City: 45 (555.00 815.00),
 '46': City: 46 (830.00 485.00),
 '47': City: 47 (1170.00 65.00),
 '48': City: 48 (830.00 610.00),
 '49': City: 49 (605.00 625.00),
 '50': City: 50 (595.00 360.00),
 '51': City: 51 (1340.00 725.00),
 '52': City: 52 (1740.00 245.00)}
```

Subsample Cities

TODO: Complete the implementation of the `subsample_cities` function in `tsp_utils.py`. The arguments are

- `cities` : the dictionary of the cities
- `number_of_cities` : the number of cities in the subsample
- `random_seed` : the random seed used to create the subsample

It should return a new dictionary of cities.

In [4]:
```
# Run

subsample_size = 10
subsample_seed = 2

cities = subsample_cities(all_cities, number_of_cities=subsample_size, random_seed=subsample_seed)

cities
```

Out[4]:
```
{'15': City: 15 (845.00 680.00),
 '26': City: 26 (1215.00 245.00),
 '40': City: 40 (760.00 650.00),
 '46': City: 46 (830.00 485.00),
 '16': City: 16 (725.00 370.00),
 '45': City: 45 (555.00 815.00),
 '48': City: 48 (830.00 610.00),
 '4': City: 4 (945.00 685.00),
 '14': City: 14 (1530.00 5.00),
 '6': City: 6 (880.00 660.00)}
```

Implement TSPNode

TODO: Complete the implementation of the `TSPNode` class. You need to implement

- `expand` This should create all possible children of this node, where a child is a swap of two neighbor cities. A state is an ordered list of city names to visit. Remember that the last city travels back to the start city and hence they are also neighbors.
- `value` This is the negative of the cost of the state. The cost of the state is the sum of the distances between the neighor cities. The distance between two neighbors is the Eucledian distance (square root of the sum of the squares of the differences).

In [5]:
```
# Run
tsp_node = TSPNode(sorted(list(cities.keys())))
tsp_node
```

Out[5]:
```
TSPNode: 14-15-16-26-4-40-45-46-48-6
```

In [6]:
```
# Run

children_nodes = tsp_node.expand()
len(children_nodes)
```

Out[6]:
```
10
```

In [7]:
```
# Run

children_nodes
```

Out[7]:
```
[TSPNode: 15-14-16-26-4-40-45-46-48-6,
 TSPNode: 14-16-15-26-4-40-45-46-48-6,
 TSPNode: 14-15-26-16-4-40-45-46-48-6,
 TSPNode: 14-15-16-4-26-40-45-46-48-6,
 TSPNode: 14-15-16-26-40-4-45-46-48-6,
 TSPNode: 14-15-16-26-4-45-40-46-48-6,
 TSPNode: 14-15-16-26-4-40-46-45-48-6,
 TSPNode: 14-15-16-26-4-40-45-48-46-6,
 TSPNode: 14-15-16-26-4-40-45-46-6-48,
 TSPNode: 6-15-16-26-4-40-45-46-48-14]
```

In [8]:
```
tsp_node.value()
```

Out[8]:
```
-4315.526779689034
```

Create a random start state

In [9]:
```
# Run
initial_seed = 5

initial_node = create_initial_node(cities, random_seed=initial_seed)

initial_node
```

Out[9]:
```
TSPNode: 4-48-26-46-40-16-15-6-45-14
```

In [10]:
```
# Run
initial_node.value()
```

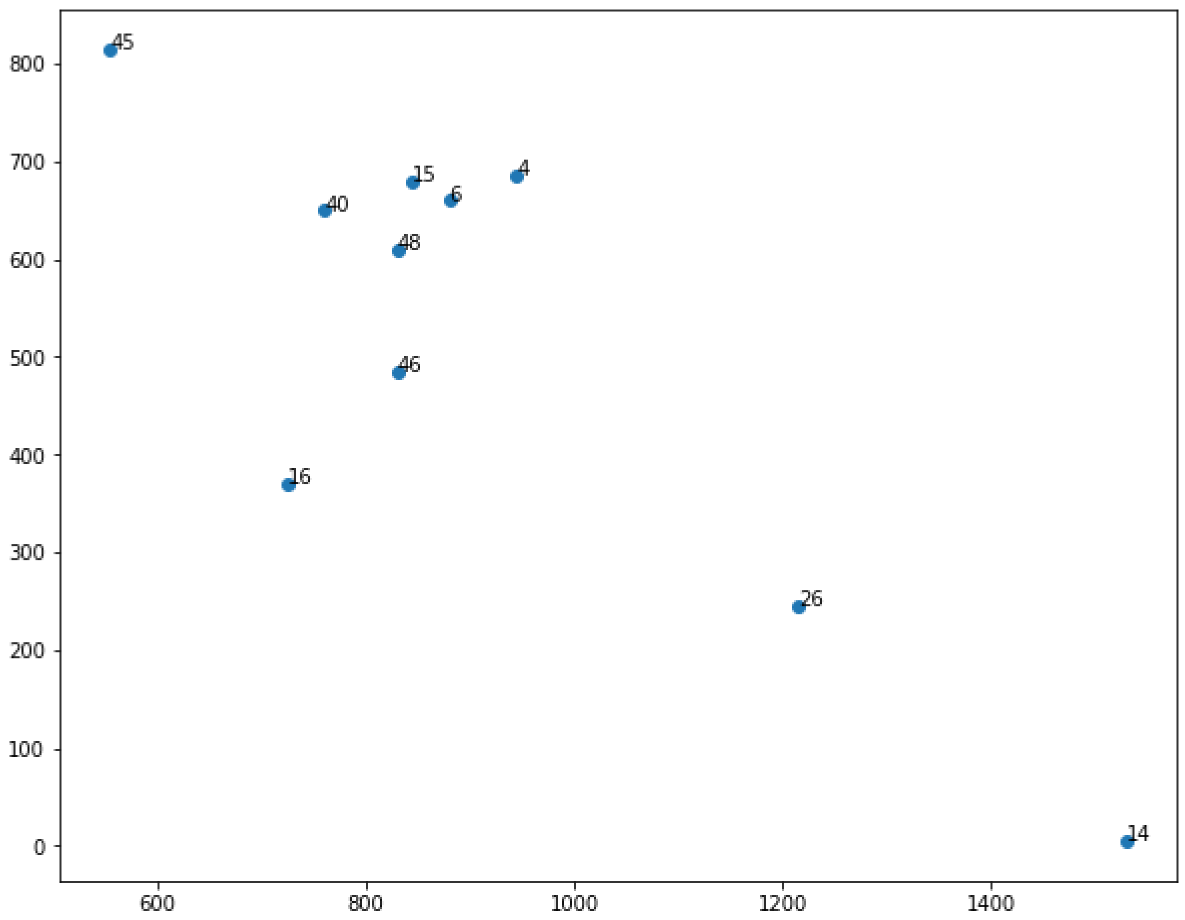
Out[10]:
```
-4480.278437200555
```

Implement Plot Functions

TODO: Implement

- `plot_cities` Given an matplotlib axes, a dictionary of cities, and a state, it should plot the cities in the state. The cities should be plotted at their coordinates.

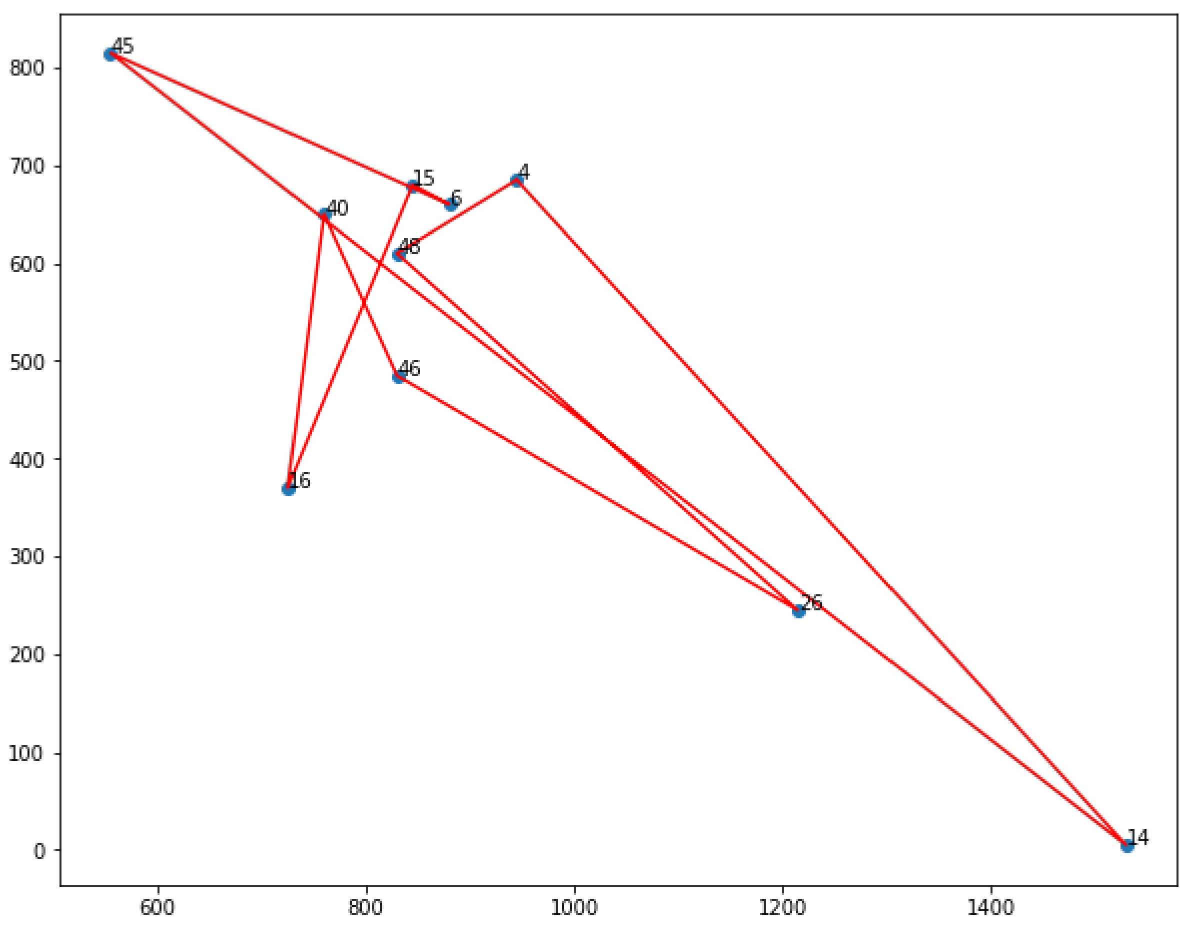
In [11]:
```
# Run
fig, ax = plt.subplots()
plot_cities(ax, all_cities, initial_node.state)
```



TODO: Implement

- `plot_path` Given an matplotlib axes, a dictionary of cities, and a state, it should plot the edges between the cities.

In [12]:
```
# Run
fig, ax = plt.subplots()
plot_cities(ax, all_cities, initial_node.state)
plot_path(ax, cities, initial_node.state)
```



Run Hill Climbing

In [13]:
```
# Run
print("Initial Node")
print(type(initial_node))
hc_sol_node = hill_climbing(initial_node)
hc_sol_node
```

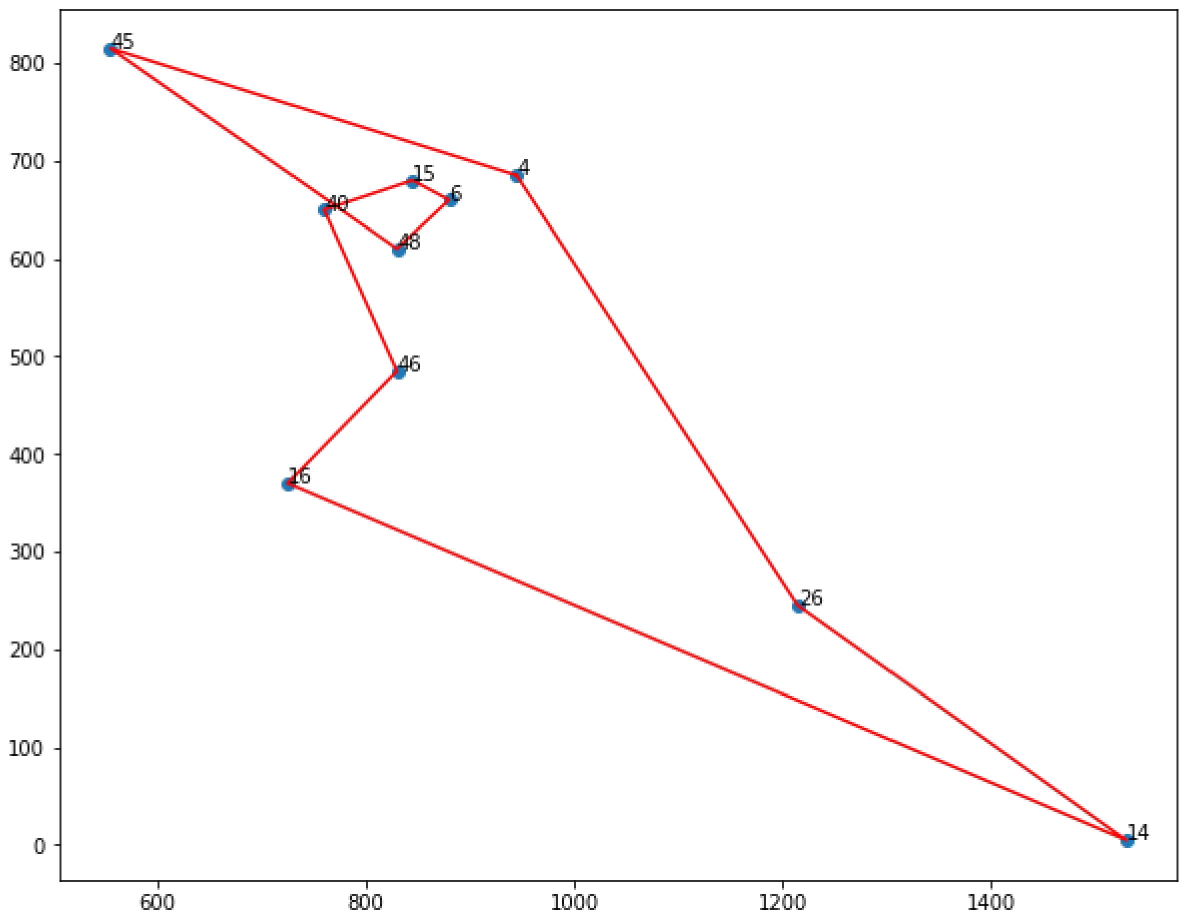
```
Initial Node
<class 'tsp_utils.TSPNode'>
```

Out[13]:
```
TSPNode: 4-26-14-16-46-40-15-6-48-45
```

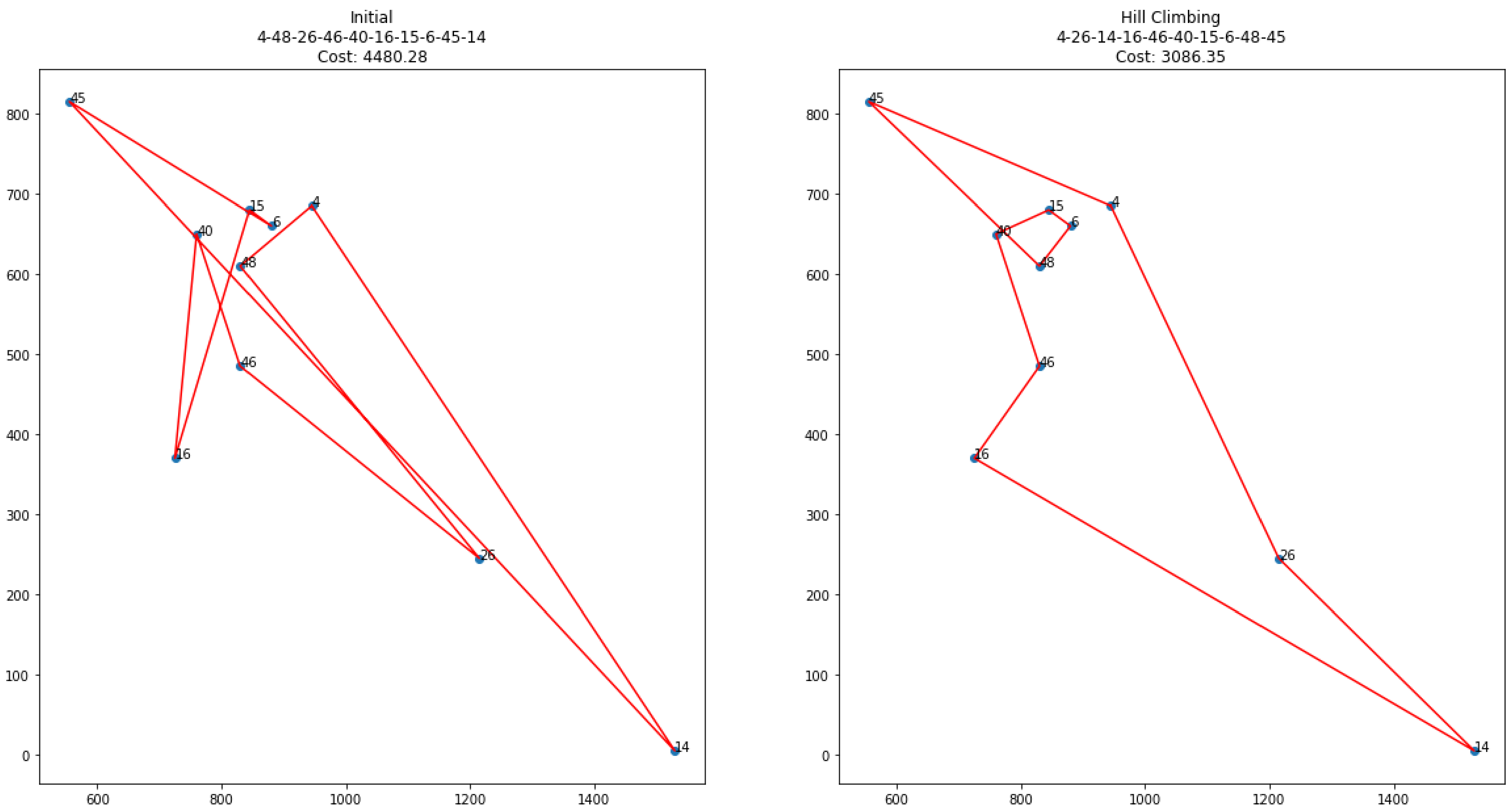
In [14]:
```
hc_sol_node.value()
```

Out[14]:
```
-3086.348120526929
```

In [15]:
```
# Plot the solution
fig, ax = plt.subplots()
plot_cities(ax, all_cities, hc_sol_node.state)
plot_path(ax, all_cities, hc_sol_node.state)
```



In [16]:
```
# Run
compare_sols((("Initial", initial_node), ("Hill Climbing", hc_sol_node)), all_cities)
```



Simulations 1 - Subsample of 10

1. Create multiple subsamples of cities, of size 10 each
2. Create multiple initializations for each.
3. Run HC for each.
4. Present the initial and the HC results as a table.

Use

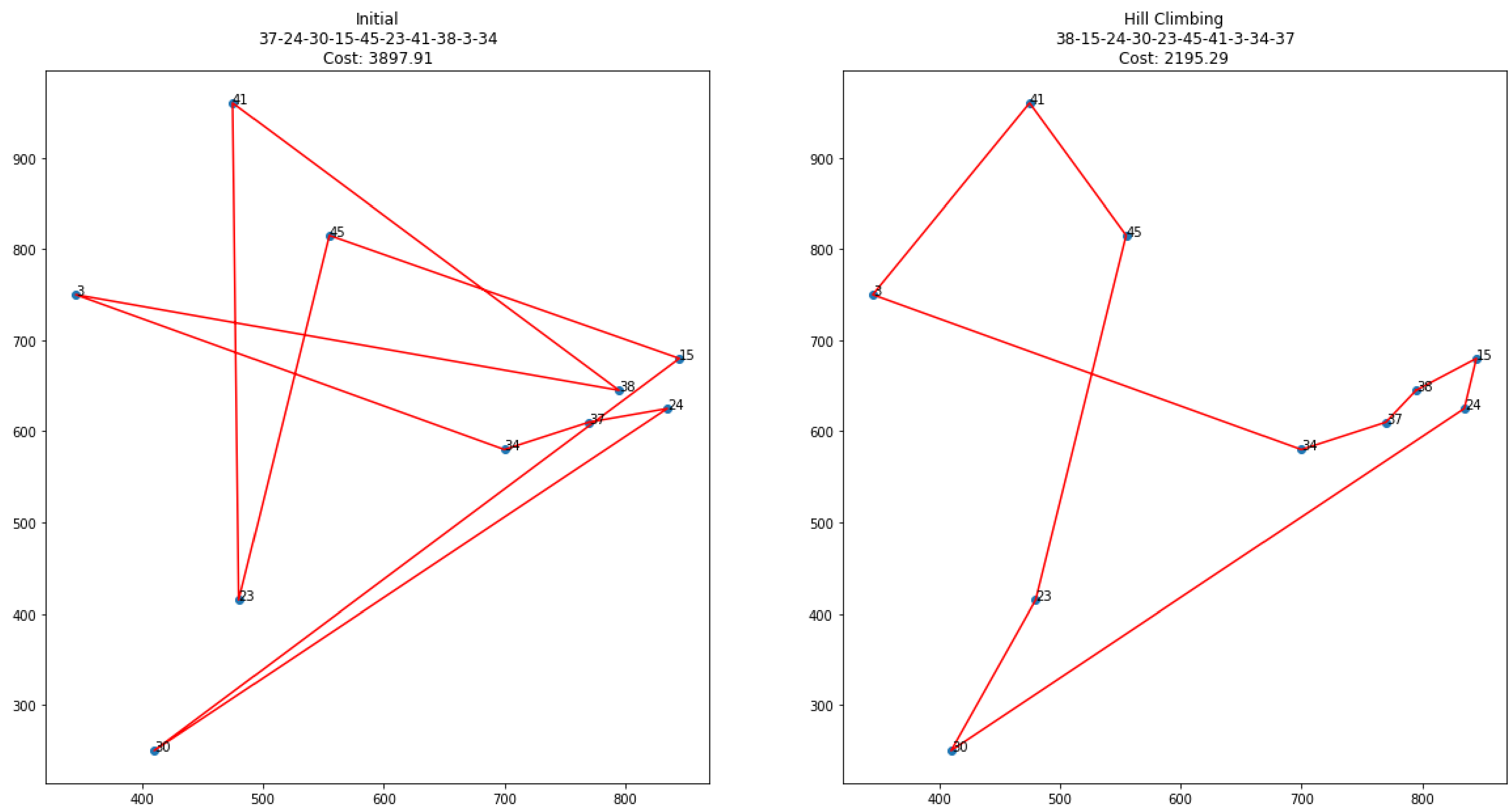
- subsample_size = 10
- subsample_seeds of `[0, 1, 2, 3, 4]`
- initial_seeds = `[11, 12, 13, 14, int(last_three_digits_of_your_CWID)]`

In [17]:
```python
# TODO - Write code and run the simulation

def run_simulations(subsample_size, subsample_seeds, initial_seeds):
    """
    Run the simulations for the given parameters.
    """
    initial_states = {}
    hc_states = {}
    for subsample_seed in subsample_seeds:
        cities = subsample_cities(all_cities, number_of_cities=subsample_size, random_seed=subsample_seed)
        initial_states[subsample_seed] = {}
        hc_states[subsample_seed] = {}
        for initial_seed in initial_seeds:
            initial_node = create_initial_node(cities, random_seed=initial_seed)
            initial_states[subsample_seed][initial_seed] = initial_node
            hc_states[subsample_seed][initial_seed] = hill_climbing(initial_node)
    return initial_states, hc_states
```

In [18]:
```python
CWID='A20473685'
subsample_size = 10
subsample_seeds = range(0, 5)
initial_seeds = [11, 12, 13, 14, 15, int(CWID[6:])]

# TODO - Complete the code. It finally should display a table.
initial_states, hc_states = run_simulations(subsample_size, subsample_seeds, initial_seeds)
import pandas as pd
df = pd.DataFrame(columns=['Subsample Seed', 'Initial Seed', 'Initial Value', 'HC Value'])
df['Subsample Seed'] = [ j for j in subsample_seeds for i in initial_seeds]
df['Initial Seed'] = [ i for j in subsample_seeds for i in initial_seeds]
df['Initial Value'] = [initial_states[j][i].value() for j in subsample_seeds for i in initial_seeds]
df['HC Value'] = [hc_states[j][i].value() for j in subsample_seeds for i in initial_seeds]
display(df)
```

	Subsample Seed	Initial Seed	Initial Value	HC Value
0	0	11	-6910.854964	-4845.283431
1	0	12	-4805.420205	-4578.810278
2	0	13	-6281.931261	-6230.336304
3	0	14	-6547.397559	-4943.598730
4	0	15	-6567.276372	-3994.034490
5	0	685	-4089.327327	-3939.512675
6	1	11	-3897.908590	-2195.289900
7	1	12	-3851.562313	-3140.493238
8	1	13	-3343.303102	-2746.034130
9	1	14	-2431.764939	-2159.997566
10	1	15	-2541.940049	-2390.651920
11	1	685	-3797.873196	-2297.324830
12	2	11	-4544.428908	-4089.549443
13	2	12	-4312.788187	-3211.747659
14	2	13	-4733.314832	-3118.205023
15	2	14	-3813.323874	-3334.947124
16	2	15	-4393.977833	-2850.467881
17	2	685	-4094.846276	-3139.174043
18	3	11	-4799.996881	-3990.085683
19	3	12	-5107.349706	-3764.067382
20	3	13	-4792.187279	-3898.438303
21	3	14	-4842.838365	-4342.710220
22	3	15	-4777.704743	-4060.666300
23	3	685	-5640.635557	-4341.438565
24	4	11	-6450.980927	-4993.819117
25	4	12	-7169.922143	-5886.325861
26	4	13	-6146.594024	-4957.369556
27	4	14	-4637.973404	-3997.536911
28	4	15	-6147.884236	-6078.827882
29	4	685	-7015.783332	-5089.433264

In [19]:
```python
# Pick subsample seed and initial seed, and visualize the initialization and the HC.

compare_sols((("Initial", initial_states[1][11]), ("Hill Climbing", hc_states[1][11])), all_cities)
```



Simulations 2

Repeat the above simulation for

- subsample of 20
- subsample of 30
- subsample of 40

Finally, repeat it for the full set of cities (i.e., no subsampling, only random initialization).

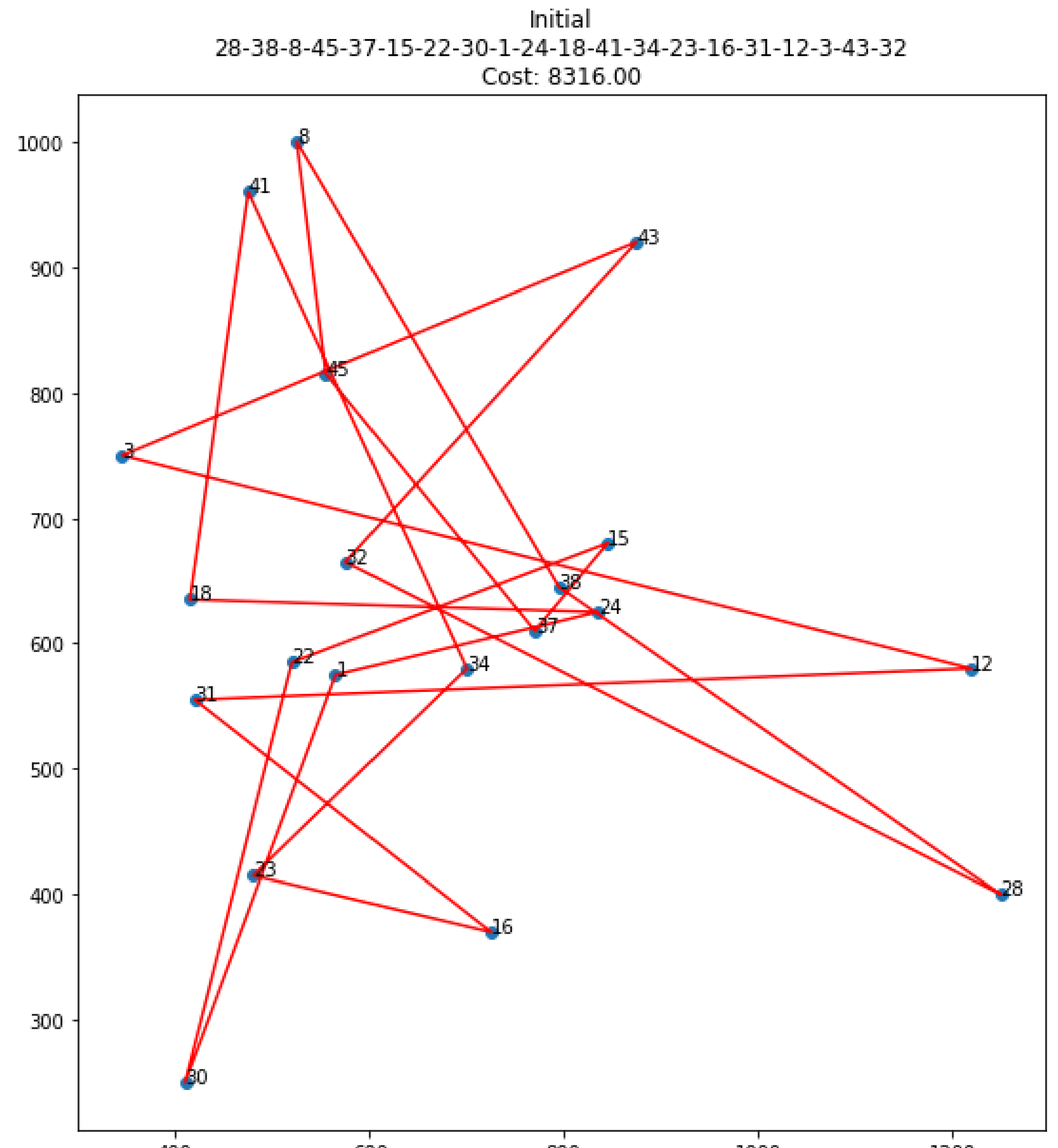
In [20]:
```
subsample_size_list=[20, 30, 40, len(all_cities)]
subsample_seeds = range(0, 5)
initial_seeds = [11, 12, 13, 14, 15, int(CWID[6:])]
for subsample_size in subsample_size_list:
    initial_states, hc_states = run_simulations(subsample_size, subsample_seeds, initial_seeds)
    df = pd.DataFrame(columns=['Subsample Seed', 'Initial Seed', 'Initial Value', 'HC Value'])
    df['Subsample Seed'] = [ j for j in subsample_seeds for i in initial_seeds]
    df['Initial Seed'] = [ i for j in subsample_seeds for i in initial_seeds]
    df['Initial Value'] = [initial_states[j][i].value() for j in subsample_seeds for i in initial_seeds]
    df['HC Value'] = [hc_states[j][i].value() for j in subsample_seeds for i in initial_seeds]
    display(df)
    compare_sols((("Initial", initial_states[1][11]), ("Hill Climbing", hc_states[1][11])), all_cities)
```

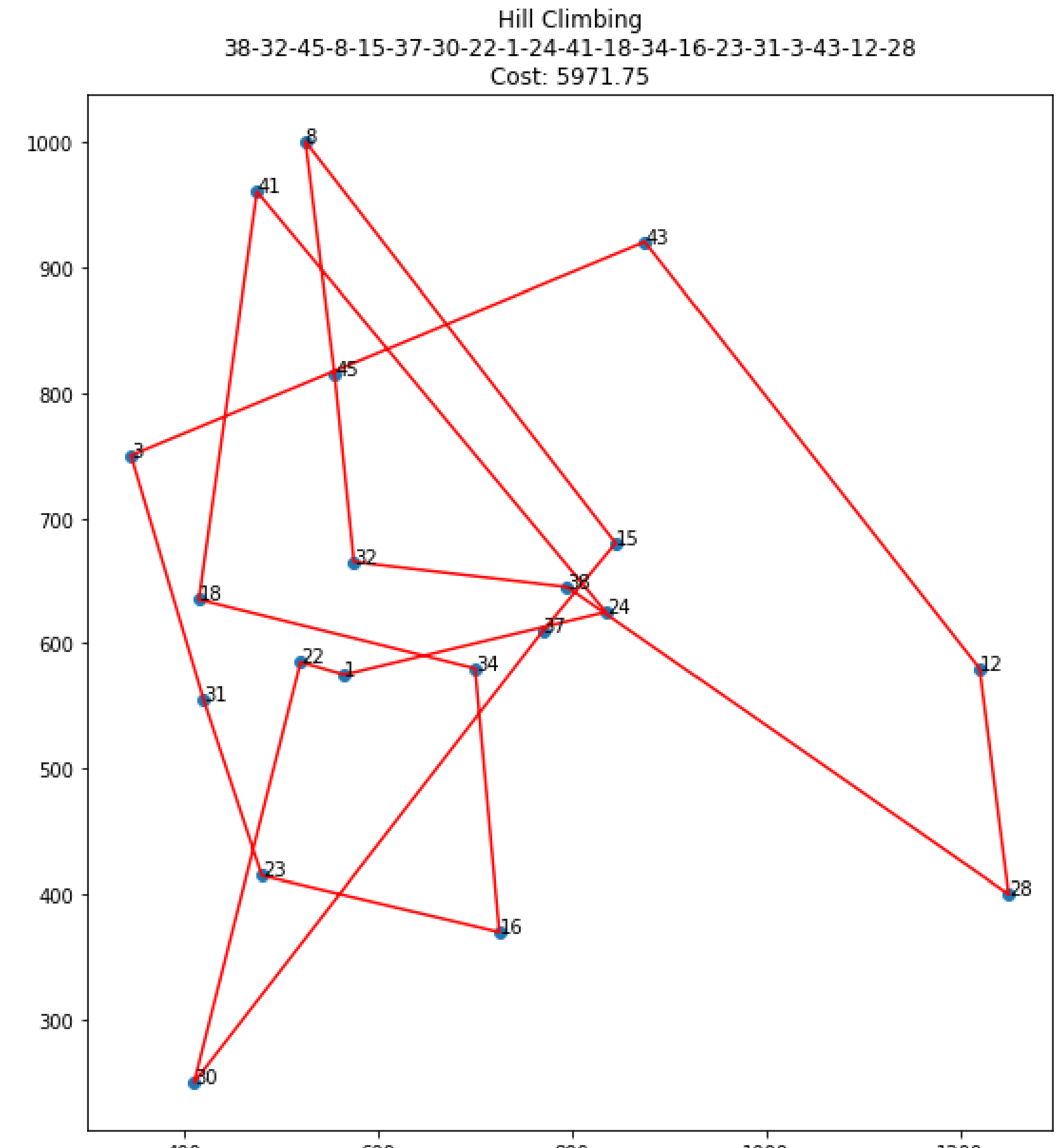
	Subsample Seed	Initial Seed	Initial Value	HC Value
0	0	11	-14528.512545	-9409.188607
1	0	12	-12492.176845	-9553.188395
2	0	13	-12938.237671	-11515.194157
3	0	14	-12529.839281	-10628.089376
4	0	15	-11961.551958	-10324.493947
5	0	685	-12764.576346	-7868.568848
6	1	11	-8315.996528	-5971.753474
7	1	12	-8106.832988	-5710.267338
8	1	13	-7919.323807	-6222.817913
9	1	14	-7841.527843	-5757.361661
10	1	15	-7579.257232	-5198.359541
11	1	685	-8142.940583	-7096.208365
12	2	11	-8682.089796	-7607.631794
13	2	12	-10238.872421	-7579.814402
14	2	13	-9250.016401	-7040.697803
15	2	14	-10050.787735	-7726.043985
16	2	15	-9214.227042	-7682.478157
17	2	685	-9768.657953	-7975.684458
18	3	11	-10567.370527	-9742.518415
19	3	12	-10543.538445	-8856.363487
20	3	13	-11736.349148	-10009.973969
21	3	14	-11166.550993	-9536.974683
22	3	15	-11551.965621	-8076.005732
23	3	685	-11500.495571	-8592.875125
24	4	11	-13671.503005	-11329.118532
25	4	12	-11587.825714	-8361.201935
26	4	13	-12308.573258	-9682.017457
27	4	14	-11053.665807	-7622.474497
28	4	15	-12269.771205	-10862.073814
29	4	685	-13194.954000	-8917.875528

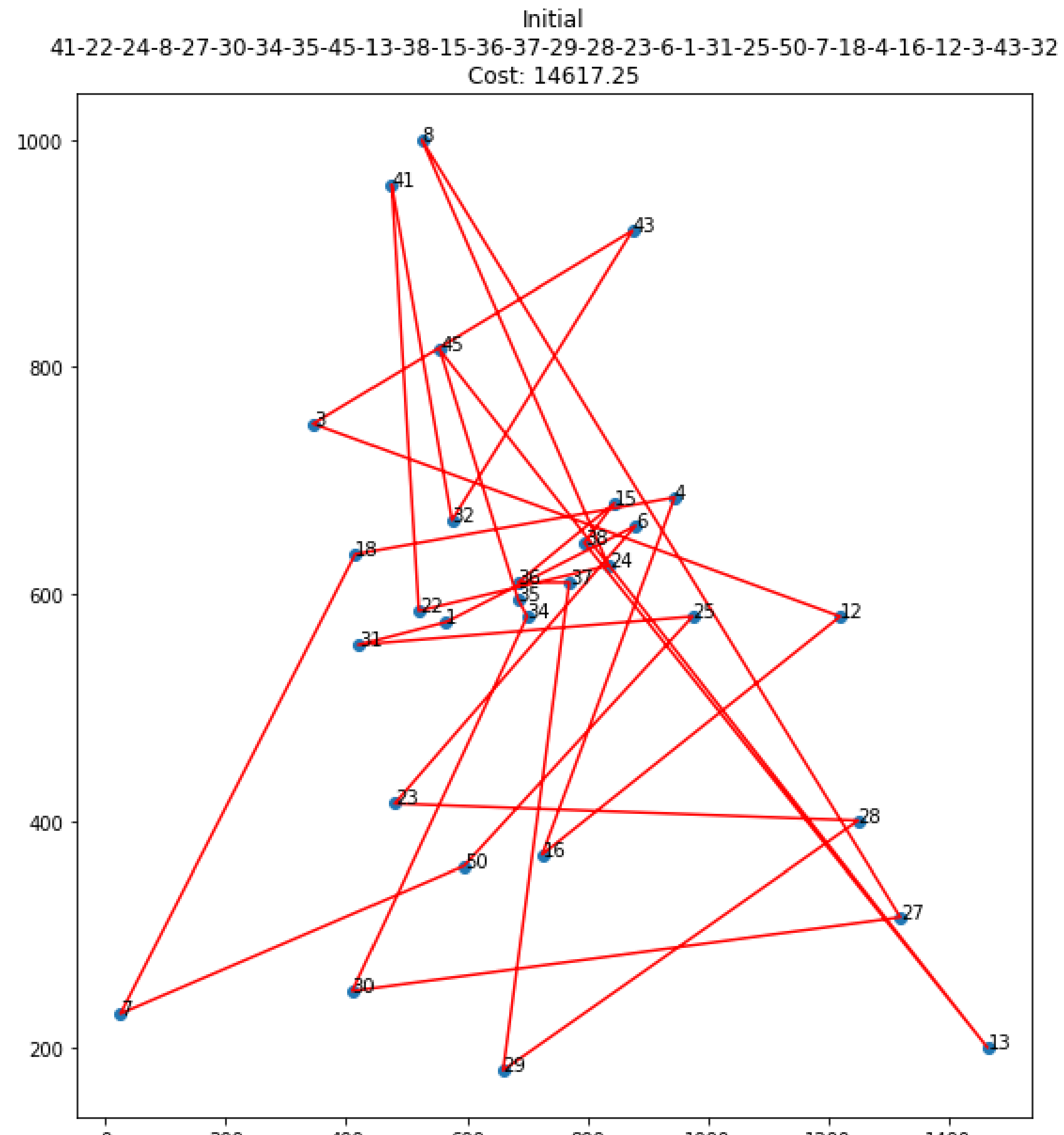
	Subsample Seed	Initial Seed	Initial Value	HC Value
0	0	11	-18452.333046	-14751.060989
1	0	12	-16392.240025	-13710.665979
2	0	13	-19073.027869	-13446.357383
3	0	14	-18543.235765	-14528.437203
4	0	15	-18072.423051	-12982.142435
5	0	685	-19757.068889	-12084.093737
6	1	11	-14617.254235	-11752.905682
7	1	12	-12924.087636	-11395.282251
8	1	13	-13329.522781	-11252.846115
9	1	14	-13921.360320	-10533.700060
10	1	15	-14768.205467	-12661.765421
11	1	685	-14854.343435	-12545.604744
12	2	11	-12965.237731	-9998.038274
13	2	12	-14217.105152	-12282.945719
14	2	13	-14082.051073	-12146.101451
15	2	14	-16210.674008	-12451.308075
16	2	15	-14568.237519	-12123.613370
17	2	685	-16189.480659	-14084.468451
18	3	11	-16658.990034	-14731.745296
19	3	12	-16427.212697	-13249.806835
20	3	13	-16750.003217	-12192.455287
21	3	14	-17842.382778	-14800.082032
22	3	15	-16098.062045	-12958.085754
23	3	685	-16440.250739	-11984.257858
24	4	11	-17191.618369	-12626.906252
25	4	12	-16373.436395	-13487.906800
26	4	13	-18476.092408	-16402.522209
27	4	14	-17528.624119	-12792.169598
28	4	15	-15100.790955	-13475.475739
29	4	685	-18562.133710	-14850.776159

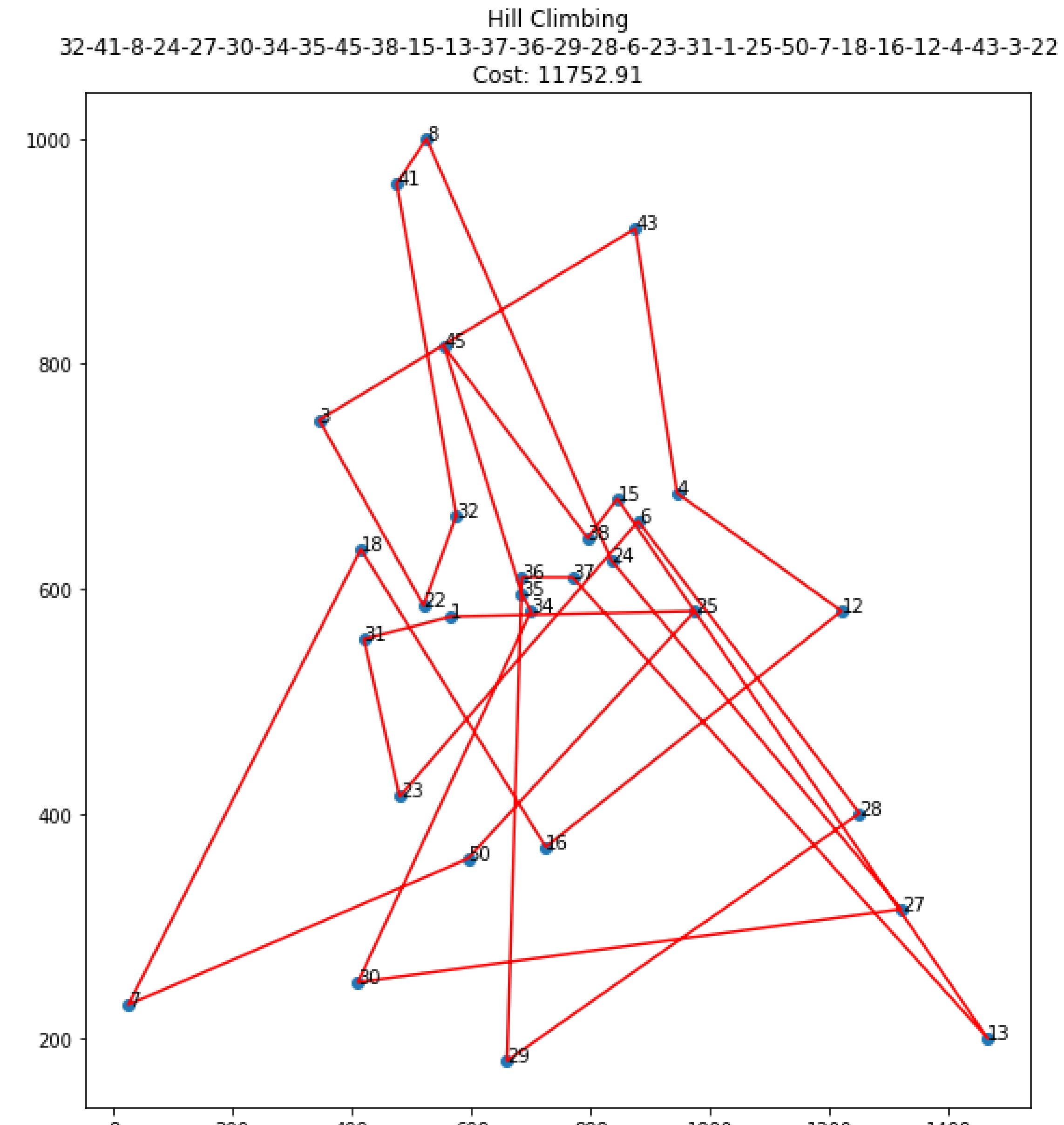
	Subsample Seed	Initial Seed	Initial Value	HC Value
0	0	11	-22422.827965	-18173.154615
1	0	12	-24004.194769	-18844.451113
2	0	13	-24314.444478	-19406.699556
3	0	14	-23950.875799	-18255.824281
4	0	15	-25058.430067	-18536.135701
5	0	685	-22871.791725	-19439.177595
6	1	11	-21039.012888	-15656.884408
7	1	12	-22582.888615	-17234.947213
8	1	13	-20314.188732	-15903.115885
9	1	14	-21356.947180	-16087.632942
10	1	15	-20338.593483	-16509.662754
11	1	685	-22128.838784	-15499.226735
12	2	11	-24160.336086	-16567.714593
13	2	12	-24050.568443	-17847.127587
14	2	13	-20854.844012	-18709.864400
15	2	14	-24817.518102	-19279.484009
16	2	15	-21517.972362	-17015.181136
17	2	685	-24082.028716	-18842.906614
18	3	11	-20334.250556	-18257.623656
19	3	12	-20291.047321	-17871.852822
20	3	13	-24315.205986	-18508.736349
21	3	14	-21786.137728	-18506.239308
22	3	15	-21698.490496	-16922.600429
23	3	685	-22193.636820	-18498.234506
24	4	11	-21728.882616	-18831.781799
25	4	12	-21626.604248	-16252.476551
26	4	13	-25140.093334	-18509.374608
27	4	14	-23924.830059	-19249.368027
28	4	15	-25000.958085	-18799.549114
29	4	685	-23818.456296	-19737.394912

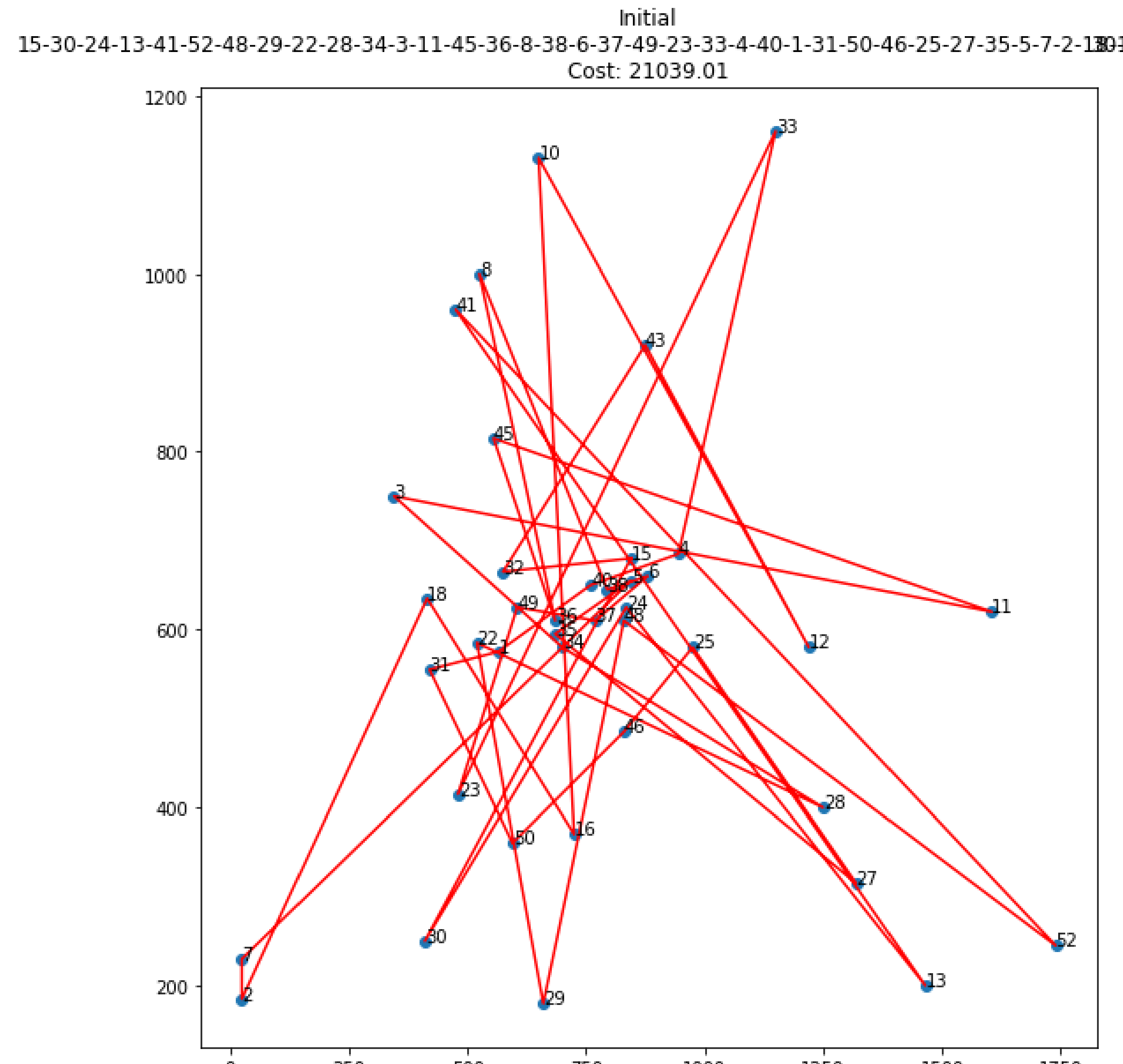
	Subsample Seed	Initial Seed	Initial Value	HC Value
0	0	11	-31341.440592	-24386.679891
1	0	12	-28501.327303	-22336.651571
2	0	13	-29191.886193	-22960.907423
3	0	14	-33497.821978	-24220.819770
4	0	15	-28864.122676	-23114.078512
5	0	685	-29767.168904	-23499.217536
6	1	11	-27733.692461	-22213.206925
7	1	12	-33503.439105	-26638.790908
8	1	13	-33253.988918	-25542.650339
9	1	14	-28666.006138	-24951.430933
10	1	15	-28543.404999	-21941.293088
11	1	685	-26764.807334	-20185.195234
12	2	11	-31283.939751	-23705.428838
13	2	12	-27940.575589	-21245.902813
14	2	13	-28888.282236	-23766.845769
15	2	14	-27127.133115	-24352.542707
16	2	15	-28939.621876	-21644.230853
17	2	685	-30799.715848	-23423.217027
18	3	11	-32396.582311	-26930.542723
19	3	12	-29835.971857	-23660.091484
20	3	13	-31618.508175	-26419.659615
21	3	14	-31006.810601	-22004.844266
22	3	15	-30719.495773	-22759.561740
23	3	685	-28248.855505	-23771.499303
24	4	11	-30332.390858	-23359.367994
25	4	12	-29153.398684	-21496.265113
26	4	13	-31219.015970	-24257.962331
27	4	14	-28269.002791	-23651.328536
28	4	15	-28666.599965	-21970.458833
29	4	685	-32505.501574	-24594.720852

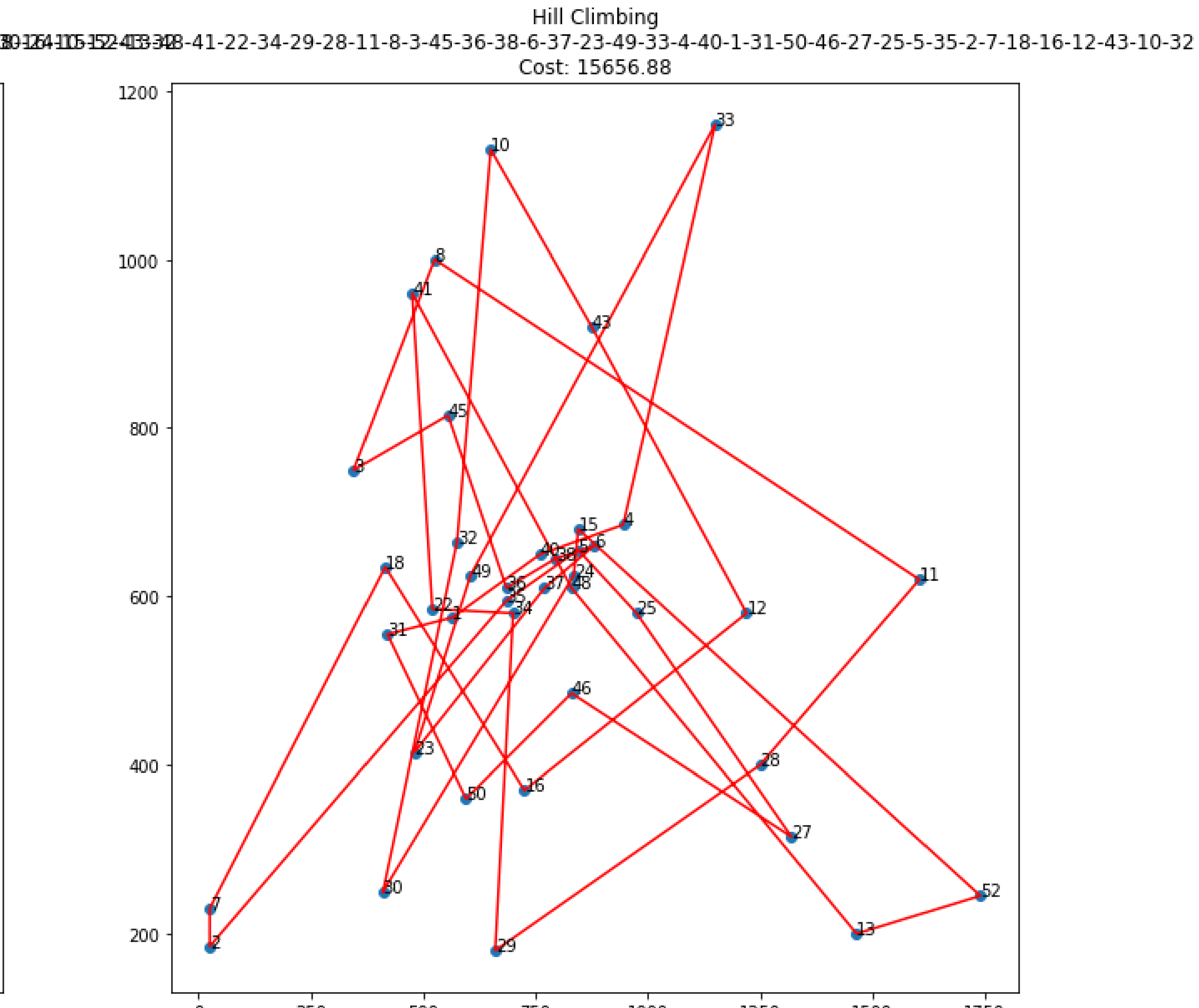

Initial
28-38-8-45-37-15-22-30-1-24-18-41-34-23-16-31-12-3-43-32
Cost: 8316.00

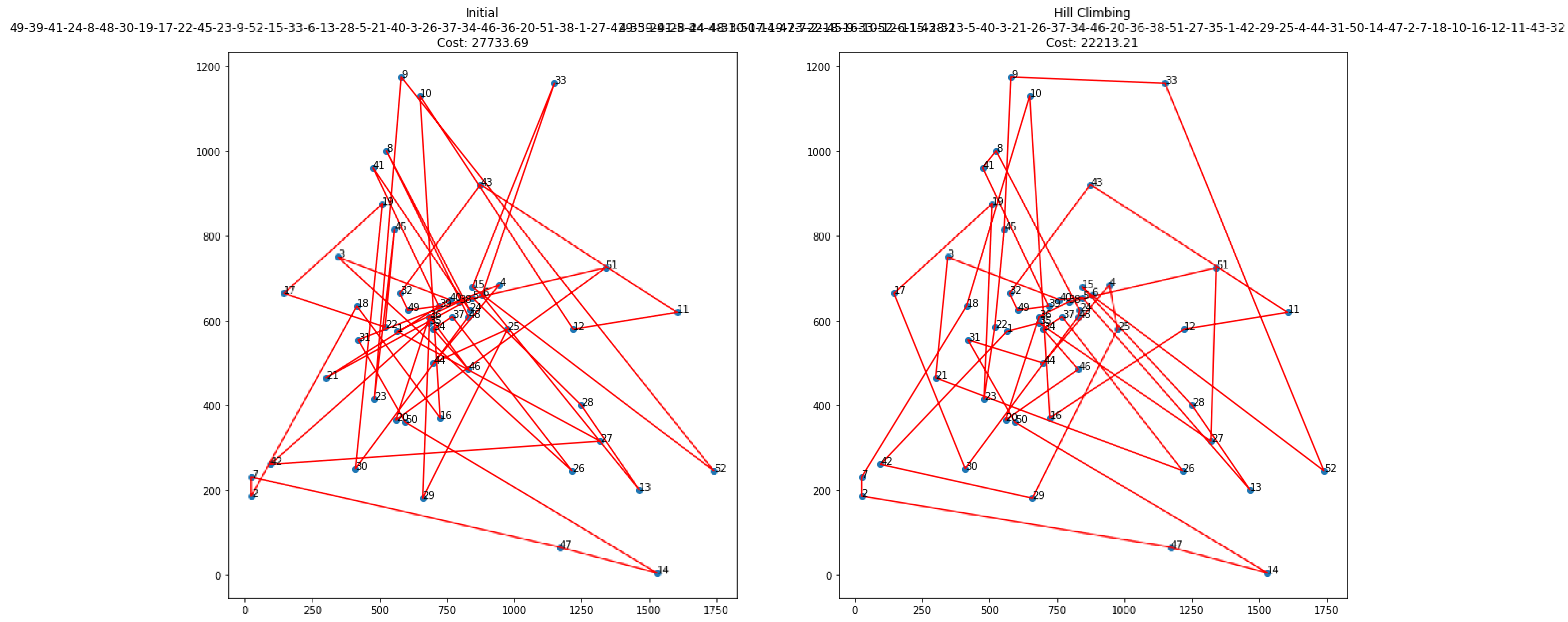

Hill Climbing
38-32-45-8-15-37-30-22-1-24-41-18-34-16-23-31-3-43-12-28
Cost: 5971.75


Initial
41-22-24-8-27-30-34-35-45-13-38-15-36-37-29-28-23-6-1-31-25-50-7-18-4-16-12-3-43-32
Cost: 14617.25


Hill Climbing
32-41-8-24-27-30-34-35-45-38-15-13-37-36-29-28-6-23-31-1-25-50-7-18-16-12-4-43-3-22
Cost: 11752.91


Initial
15-30-24-13-41-52-48-29-22-28-34-3-11-45-36-8-38-6-37-49-23-33-4-40-1-31-50-46-25-27-35-5-7-2
Cost: 21039.01


Hill Climbing
41-22-34-29-28-11-8-3-45-36-38-6-37-23-49-33-4-40-1-31-50-46-27-25-5-35-2-7-18-16-12-43-10-32
Cost: 15656.88



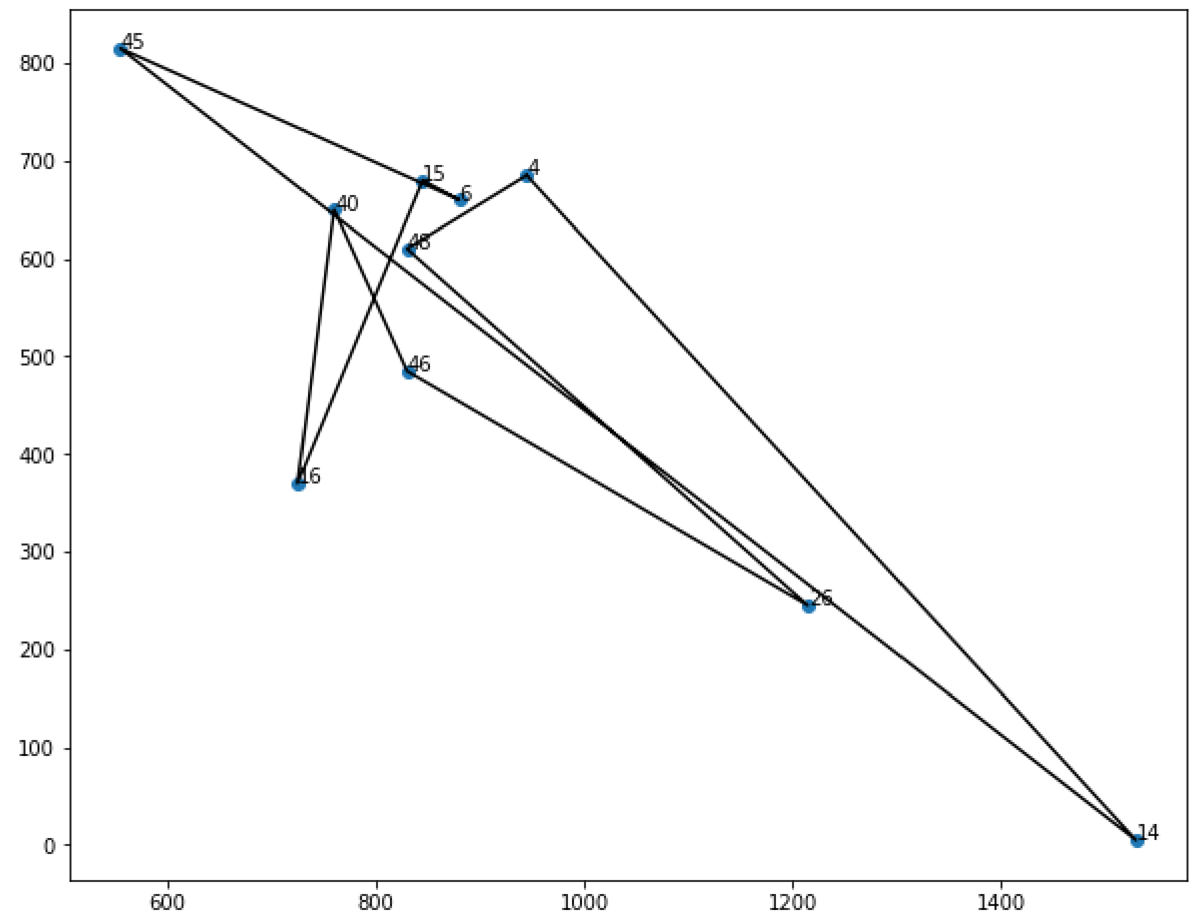
Optional (for fun only - no extra credit)

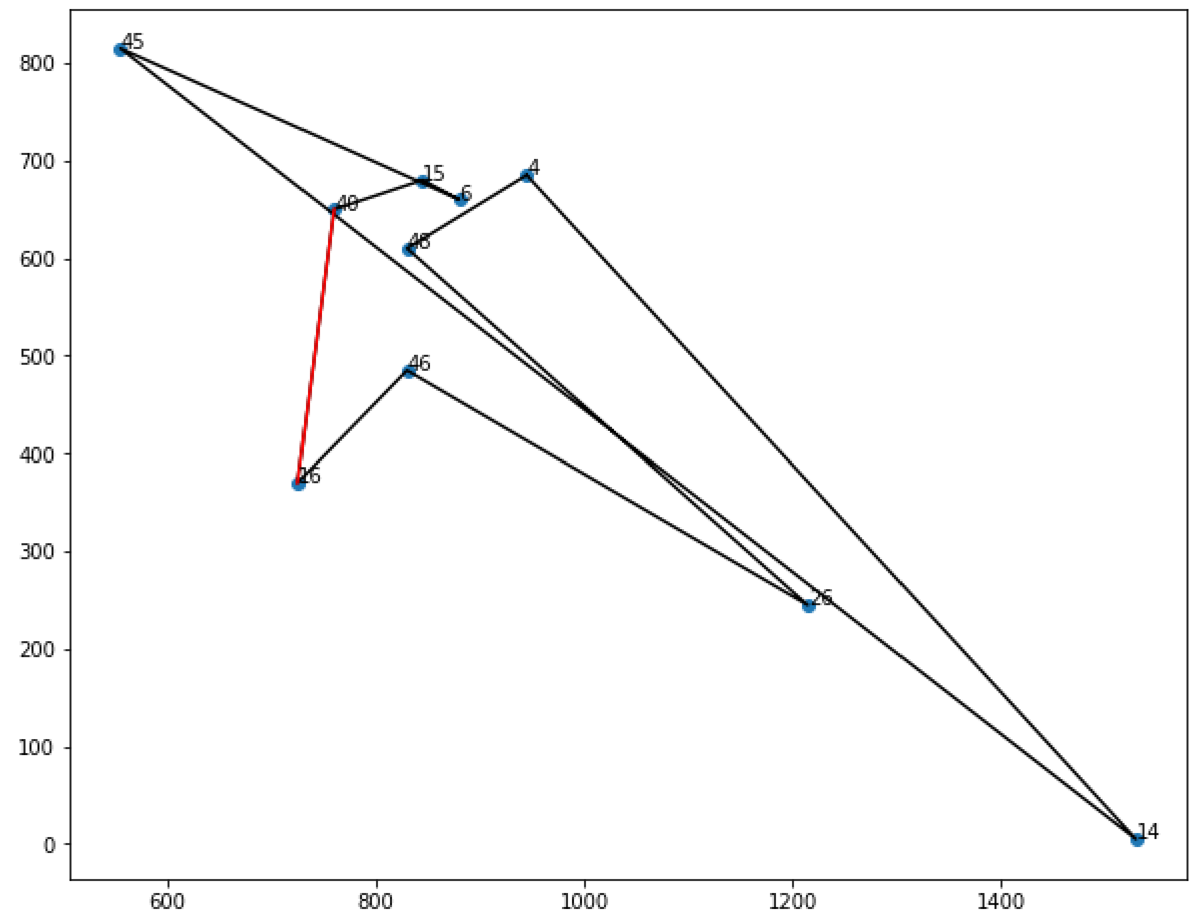
Given a Hill Climbing solution, trace the path to the initial state (using the `path()` function) and visualize the differences between each successsor state.

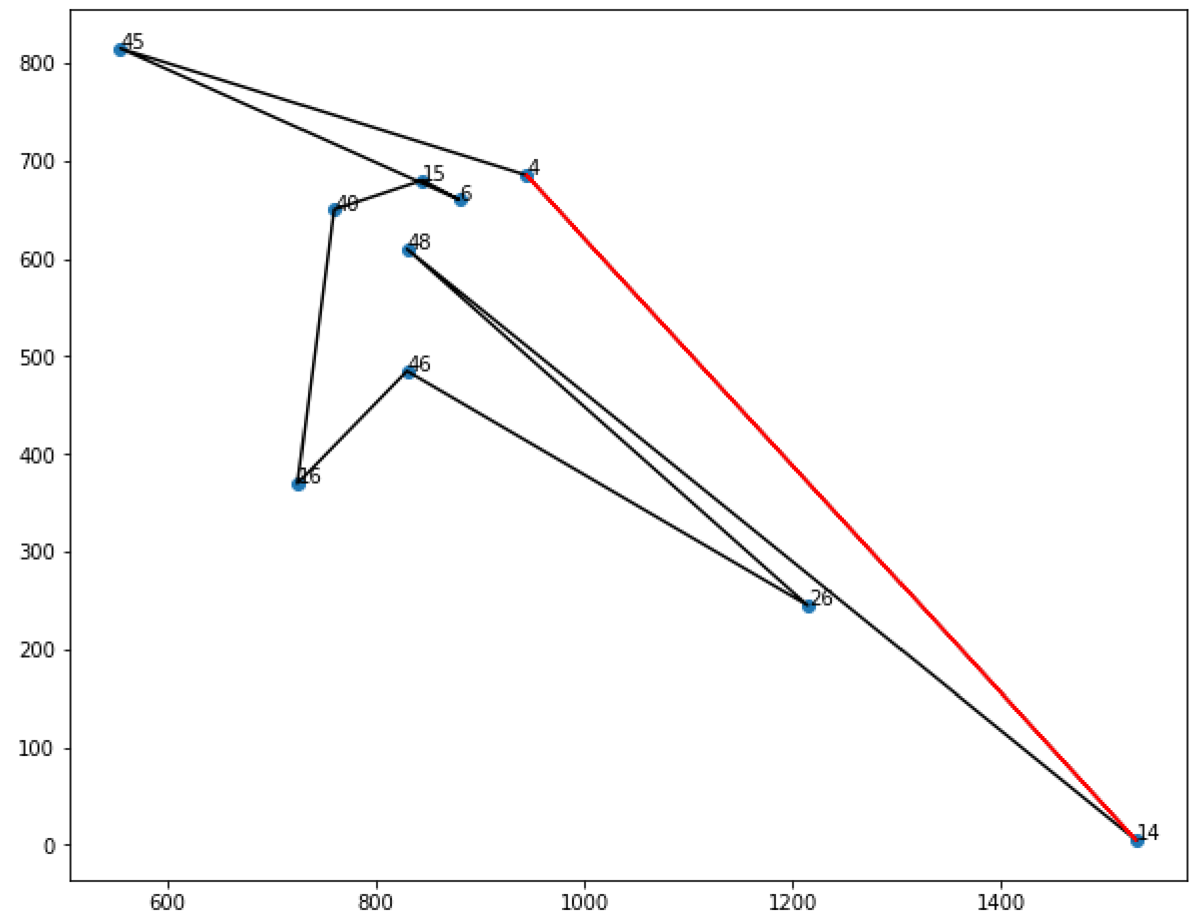
Here is a simple one. I'm sure you can come up with fancier ones.

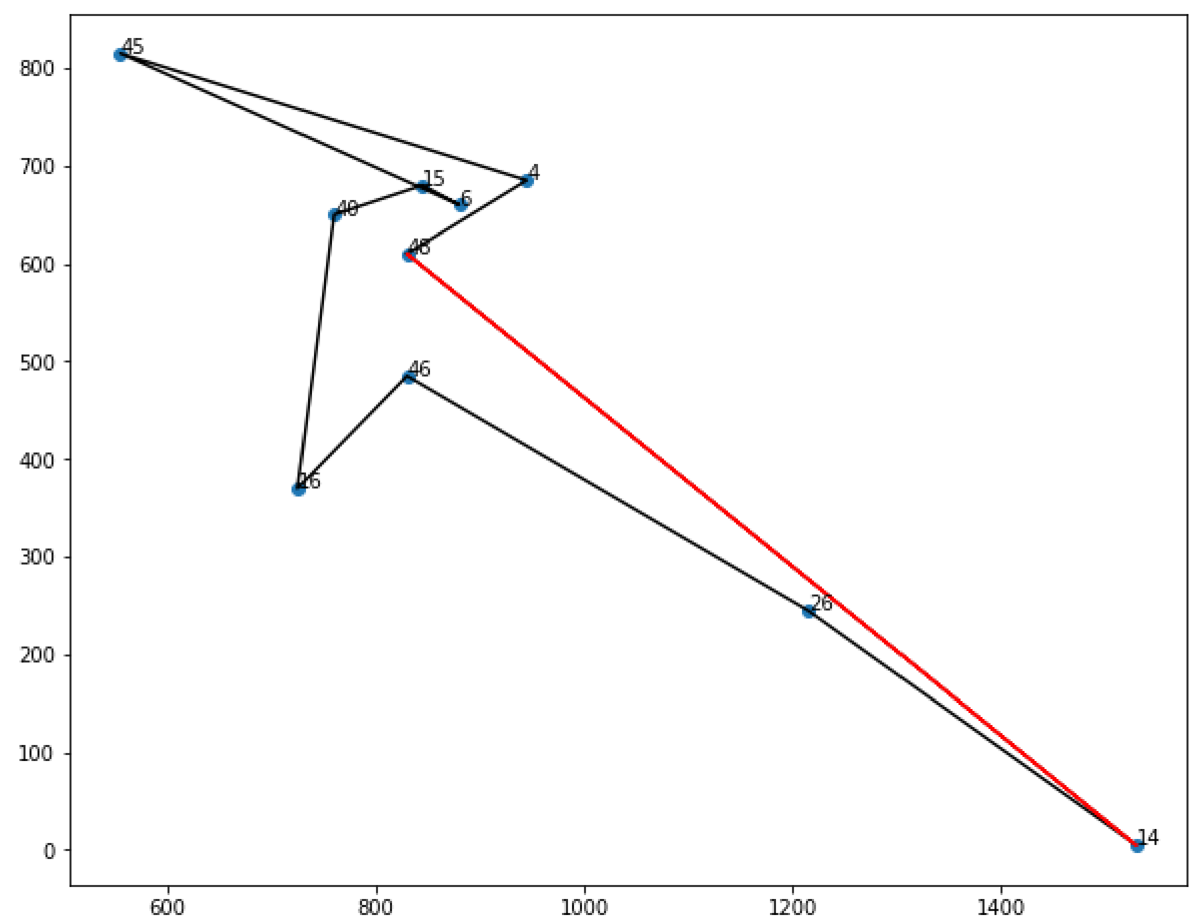
In [21]:
```
def plot_path_diff(cities, state, path):
    """
    Plot the path differences.
    """

    path=[i for i in path]
    xtmp=[]
    ytmp=[]
    for i in range(len(path) - 1):

        fig, ax = plt.subplots()
        state1 = path[i].state
        state2 = path[i + 1].state
        state=state1
        x = [cities[i].x for i in state]
        y = [cities[i].y for i in state]

        ax.scatter(x, y)
        for j in range(len(state)):
            ax.annotate(state[j], (x[j], y[j]))
        x+=x[0:1]
        y+=y[0:1]
        ax.plot(x, y, color='black')
        if i==0:
            xtmp=x
            ytmp=y
        else:
            xtmp_2=[xtmp[j] if x[j]!=xtmp[j] else 0 for j in range(len(x))]
            ytmp_2=[ytmp[j] if y[j]!=ytmp[j] else 0 for j in range(len(y))]
            xtmp=x
            ytmp=y
            xtmp_2=[xtmp_2[j] for j in range(len(xtmp_2)) if xtmp_2[j]!=0]
            ytmp_2=[ytmp_2[j] for j in range(len(ytmp_2)) if ytmp_2[j]!=0]
            #print(xtmp_2,ytmp_2)
            ax.plot(xtmp_2, ytmp_2, color='red')
```

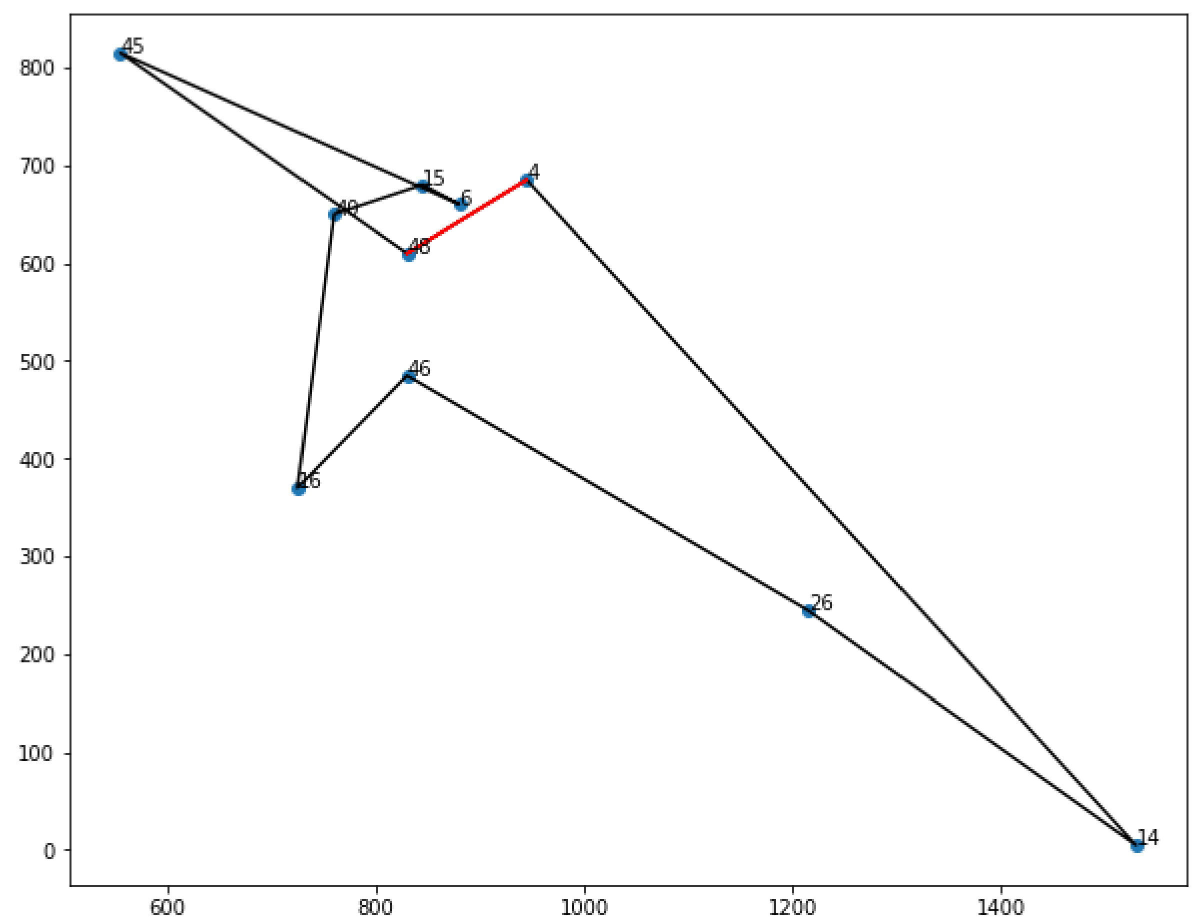
In [22]:
```
plot_path_diff(all_cities, hc_sol_node.state, hc_sol_node.path())
```

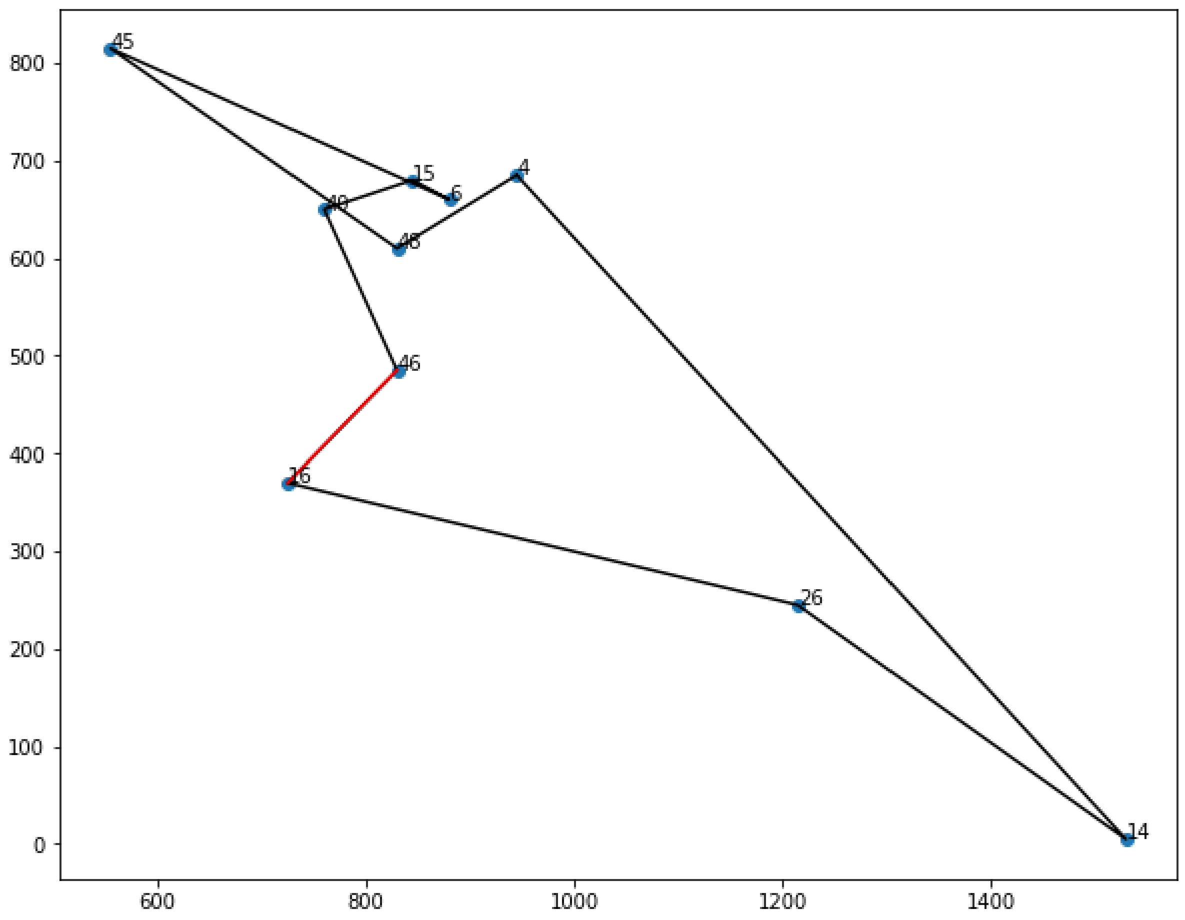


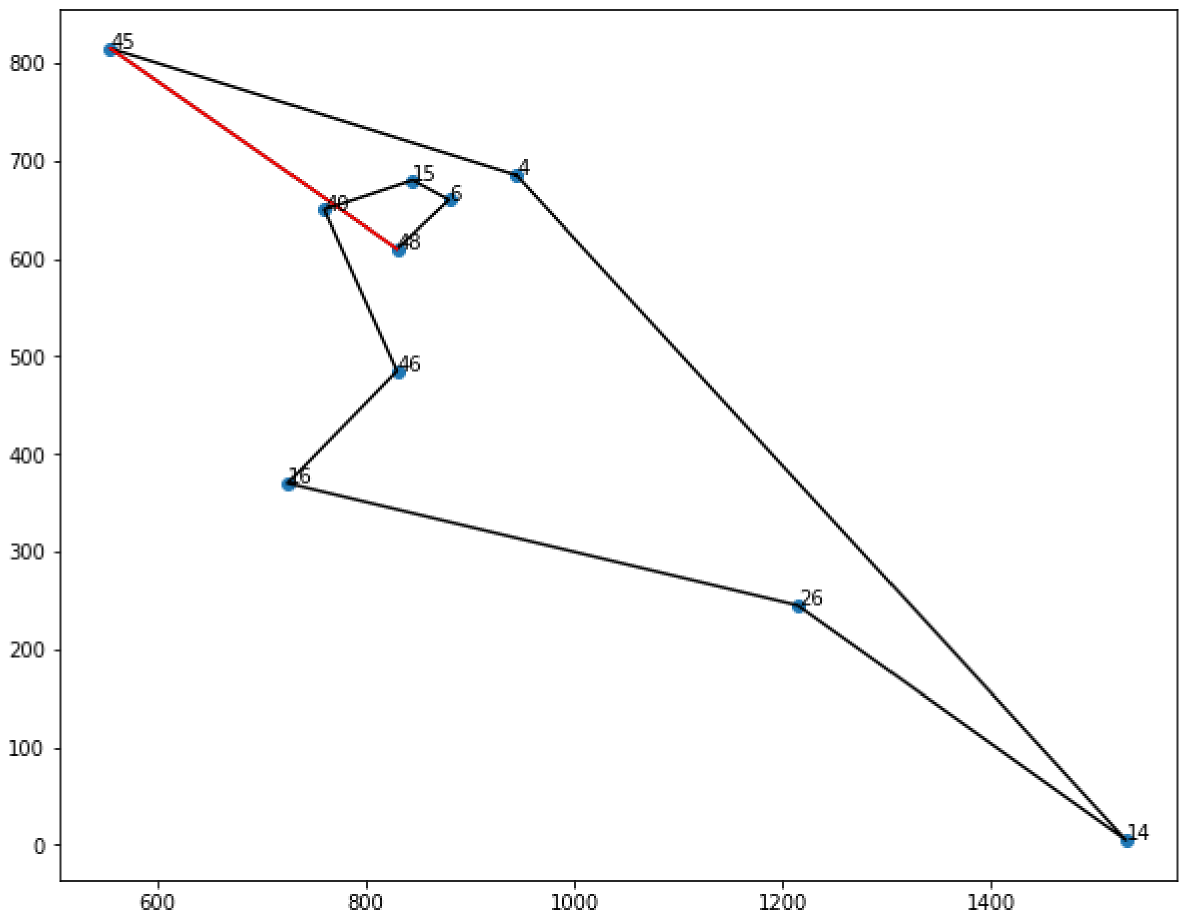

45
15
6
4
40
48
46
16
26
14


45
4
15
6
40
48
46
16
26
14


45
15
4
6
40
48
46
16
26
14


45
15
4
6
40
48
46
16
26
14





In [ ]: