shortPath nb

April 29, 2018

In [1]:
```
# Austin Griffith
# Python 3.6.5
# 4/25/2018

import pandas as pd
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import networkx as nx
```

In [2]:
```
# set up plotting parameters
params = {'legend.fontsize': 20,
          'figure.figsize': (13,9),
         'axes.labelsize': 20,
         'axes.titlesize':20,
         'xtick.labelsize':15,
         'ytick.labelsize':15}
pylab.rcParams.update(params)
```

In [3]:
```
# graph all nodes and paths
def networkCompletePlot(solution,maxNode):
    G = nx.DiGraph()
    G.add_nodes_from(range(0,maxNode+1))
    for i,j in nodes:
        G.add_edge(i,j)

    # get solution nodes
    sp = [i for i,j in solution[1]]
    sp.append(end)

    colorNode = ['white' if not node in sp else 'red' for node in G.nodes()]
    title = 'Complete Network: Gamma = '+str(int(solution[0]))+', Opt Obj = '+str(round(
    nx.draw_networkx(G,node_color=colorNode,node_size=200)
    plt.axis('off')
    plt.title(title)
    plt.show()
```

```
# graph path, with costs on edges
def networkPathPlot(solution,maxNode,cost):
    # get solution nodes
    sp = [i for i,j in solution[1]]
    sp.append(end)

    # set up random position values
    a = np.arange(maxNode+1)
    b = np.arange(maxNode+1)
    np.random.shuffle(a)
    posArray = np.array([a,b]).transpose()

    positions = {}
    for p in range(0,len(sp)):
        L = posArray[p]
        positions[sp[p]] = (L[0],L[1])

    # set up network graph
    G = nx.DiGraph()
    G.add_nodes_from(sp)

    for i,j in tuplelist(solution[1]):
        G.add_edge(i,j)

    labels = {}
    for i in solution[1]:
        labels[i] = round(c[i],3)

    title = 'Optimal Path: Gamma = '+str(int(solution[0]))+', Opt Obj = '+str(round(solu
    nx.draw_networkx(G,positions,node_size=350)
    nx.draw_networkx_edge_labels(G,positions,edge_labels=labels)
    plt.axis('off')
    plt.title(title)
    plt.show()
```

In [4]:
```
# pull data
edges = pd.read_csv('edge_data.csv')
edges['i'] = np.int64(edges['i'])
edges['j'] = np.int64(edges['j'])

# create dictionaries of edge values
c = {}
d = {}
nodes = tuplelist()
for i in edges.index:
    c[edges['i'][i],edges['j'][i]] = edges['c(ij)'][i]
    d[edges['i'][i],edges['j'][i]] = edges['d(ij)'][i]
    nodes.append((edges['i'][i],edges['j'][i]))
```

```
maxNodes = max(edges['j'])
minNodes = min(edges['i'])
```

In [5]:
```
# choose start and end nodes
start = 0
end = 49

# allowed edge congestions
gend = 4
gammas = np.linspace(0,gend,gend+1)
print('Allowed Congestions:')
print(gammas)
```

```
Allowed Congestions:
[ 0.  1.  2.  3.  4.]
```

In [6]:
```
# initialize model
model = Model('Shortest_Path')

# set up x binary variables, set to each location/movement
xVars = model.addVars(nodes, vtype=GRB.BINARY, name='move')
y0 = model.addVar(vtype=GRB.CONTINUOUS, name='y0')
zVars = model.addVars(nodes, lb=0.0, vtype=GRB.CONTINUOUS, name='cong')
model.update()
```

In [7]:
```
# constrain all entrance and exit nodes
enterStart = []
leaveStart = []
enterEnd = []
leaveEnd = []
for n in nodes:
    # for start nodes
    if n[0] == start:
        leaveStart.append(xVars[n])
    elif n[1] == start:
        enterStart.append(xVars[n])
    # for end nodes
    if n[0] == end:
        leaveEnd.append(xVars[n])
    elif n[1] == end:
        enterEnd.append(xVars[n])

model.addConstr(quicksum(leaveStart) == 1)
model.addConstr(quicksum(enterStart) == 0)
model.addConstr(quicksum(leaveEnd) == 0)
model.addConstr(quicksum(enterEnd) == 1)
model.update()
```

In [8]:
```
# gather all paths
paths = []
for i in range(minNodes+1,maxNodes):
    pathFrom = []
    pathTo = []
    for n in nodes:
        if n[0] == i:
            pathFrom.append(xVars[n])
        elif n[1] == i:
            pathTo.append(xVars[n])
    paths.append([pathFrom,pathTo])
model.update()

for p in paths:
    model.addConstr(quicksum(p[0]) - quicksum(p[1]) == 0.0)
model.update()

print('Example of Path Constraint for a Given Node:')
print(quicksum(p[0]) - quicksum(p[1]))
```

```
Example of Path Constraint for a Given Node:
<gurobi.LinExpr: move[48,0] + move[48,1] + move[48,2] + move[48,3] + move[48,4] + move[48,5] + m
```

In [9]:
```
# objective function
costObj = []
for n in nodes:
    costObj.append(xVars[n]*c[n])
    model.addConstr(zVars[n] >= xVars[n]*d[n] - y0)
model.update()

print('Example of Congestion Constraint:')
print(zVars[n],' >= ',xVars[n]*d[n] - y0)
```

```
Example of Congestion Constraint:
<gurobi.Var cong[49,48]>  >=  <gurobi.LinExpr: 3.377703354 move[49,48] + -1.0 y0>
```

In [10]:
```
# iterate optimization through various gammas (congestions)
output = []
for g in gammas:
    # optimize
    objective = quicksum(costObj) + g*y0 + quicksum(zVars)
    model.setObjective(objective, GRB.MINIMIZE)

    model.optimize()

    # order the printout of optimal edges
    moves = []
```

```python
            for m in xVars:
                if xVars[m].x != 0:
                    moves.append(m)
            order = [moves[0]]
            for i in range(len(moves)):
                for m in moves:
                    if order[i][1] == m[0]:
                        order.append(m)
            output.append([g,order,model.objVal])
```

```
Optimize a model with 2261 rows, 4419 columns and 11045 nonzeros
Variable types: 2210 continuous, 2209 integer (2209 binary)
Coefficient statistics:
  Matrix range     [1e-03, 5e+00]
  Objective range  [1e-04, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 0.0648983
Presolve removed 2261 rows and 4419 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 8 available processors)

Solution count 1: 0.0648983

Optimal solution found (tolerance 1.00e-04)
Best objective 6.489829400000e-02, best bound 6.489829400000e-02, gap 0.0000%
Optimize a model with 2261 rows, 4419 columns and 11045 nonzeros
Variable types: 2210 continuous, 2209 integer (2209 binary)
Coefficient statistics:
  Matrix range     [1e-03, 5e+00]
  Objective range  [1e-04, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]

Loaded MIP start with objective 5.0649

Presolve removed 161 rows and 1217 columns
Presolve time: 0.12s
Presolved: 2100 rows, 3202 columns, 18797 nonzeros
Found heuristic solution: objective 5.0569050
Variable types: 1164 continuous, 2038 integer (2038 binary)

Root relaxation: objective 6.301364e-01, 150 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
```

```
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.63014    0   50    5.05690    0.63014  87.5%     -    0s
H    0     0                         1.4241510    0.63014  55.8%     -    0s
H    0     0                         1.4195547    0.63014  55.6%     -    0s
H    0     0                         1.2783774    0.63014  50.7%     -    0s
     0     0    0.91670    0   20    1.27838    0.91670  28.3%     -    0s
     0     0    1.08236    0   14    1.27838    1.08236  15.3%     -    0s
     0     0    1.16091    0    7    1.27838    1.16091  9.19%     -    0s

Cutting planes:
  MIR: 1

Explored 1 nodes (255 simplex iterations) in 0.27 seconds
Thread count was 8 (of 8 available processors)

Solution count 4: 1.27838 1.41955 1.42415 5.0569

Optimal solution found (tolerance 1.00e-04)
Best objective 1.278377432000e+00, best bound 1.278377432000e+00, gap 0.0000%
Optimize a model with 2261 rows, 4419 columns and 11045 nonzeros
Variable types: 2210 continuous, 2209 integer (2209 binary)
Coefficient statistics:
  Matrix range     [1e-03, 5e+00]
  Objective range  [1e-04, 2e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]

Loaded MIP start with objective 1.53292

Presolve removed 91 rows and 178 columns
Presolve time: 0.01s
Presolved: 2170 rows, 4241 columns, 10600 nonzeros
Variable types: 2121 continuous, 2120 integer (2120 binary)

Root relaxation: objective 8.840013e-01, 65 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.88400    0   43    1.53292    0.88400  42.3%     -    0s
     0     0    1.13083    0   23    1.53292    1.13083  26.2%     -    0s
     0     0    1.13083    0   19    1.53292    1.13083  26.2%     -    0s
     0     0    1.21769    0   34    1.53292    1.21769  20.6%     -    0s
     0     0    1.21769    0   18    1.53292    1.21769  20.6%     -    0s
     0     0    1.28855    0   18    1.53292    1.28855  15.9%     -    0s
     0     0    1.28855    0    5    1.53292    1.28855  15.9%     -    0s
H    0     0                         1.3536473    1.28855  4.81%     -    0s
```

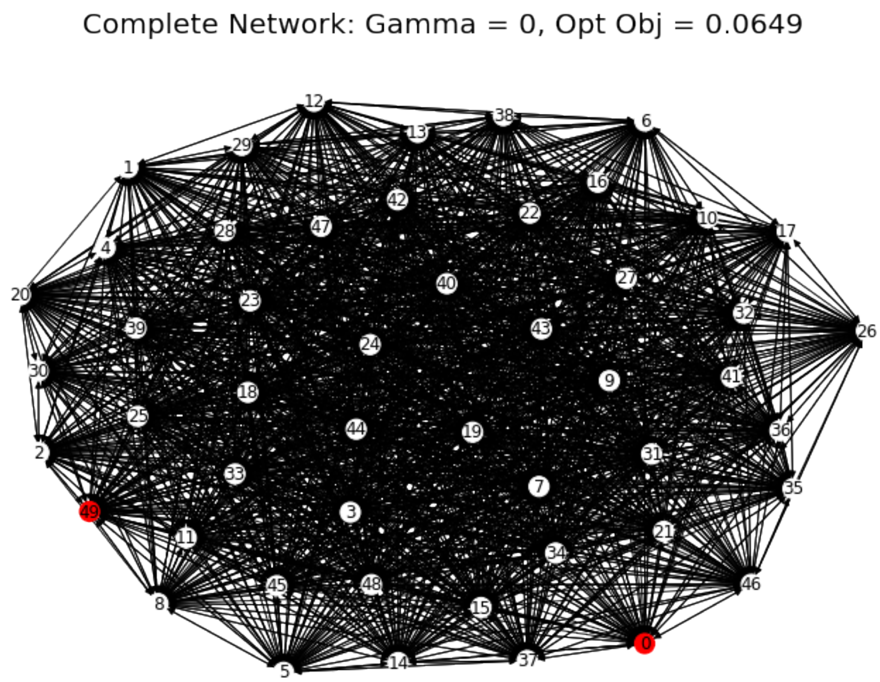
```
H    0     0                       1.3387639    1.28855  3.75%     -    0s
     0     0    1.33521    0    5    1.33876    1.33521  0.27%     -    0s

Explored 1 nodes (268 simplex iterations) in 0.12 seconds
Thread count was 8 (of 8 available processors)

Solution count 3: 1.33876 1.35365 1.53292

Optimal solution found (tolerance 1.00e-04)
Best objective 1.338763884000e+00, best bound 1.338763884000e+00, gap 0.0000%
Optimize a model with 2261 rows, 4419 columns and 11045 nonzeros
Variable types: 2210 continuous, 2209 integer (2209 binary)
Coefficient statistics:
  Matrix range     [1e-03, 5e+00]
  Objective range  [1e-04, 3e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]

Loaded MIP start with objective 1.39286

Presolve removed 91 rows and 178 columns
Presolve time: 0.01s
Presolved: 2170 rows, 4241 columns, 10600 nonzeros
Variable types: 2121 continuous, 2120 integer (2120 binary)

Root relaxation: objective 1.000748e+00, 63 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    1.00075    0   36    1.39286    1.00075  28.2%     -    0s
     0     0    1.31029    0   12    1.39286    1.31029  5.93%     -    0s
     0     0    1.31712    0    8    1.39286    1.31712  5.44%     -    0s
H    0     0                       1.3872194    1.31712  5.05%     -    0s
H    0     0                       1.3387639    1.31712  1.62%     -    0s
     0     0     cutoff    0         1.33876    1.33876  0.00%     -    0s

Cutting planes:
  Gomory: 1
  Implied bound: 2

Explored 1 nodes (119 simplex iterations) in 0.08 seconds
Thread count was 8 (of 8 available processors)

Solution count 3: 1.33876 1.38722 1.39286

Optimal solution found (tolerance 1.00e-04)
Best objective 1.338763884000e+00, best bound 1.338763884000e+00, gap 0.0000%
```

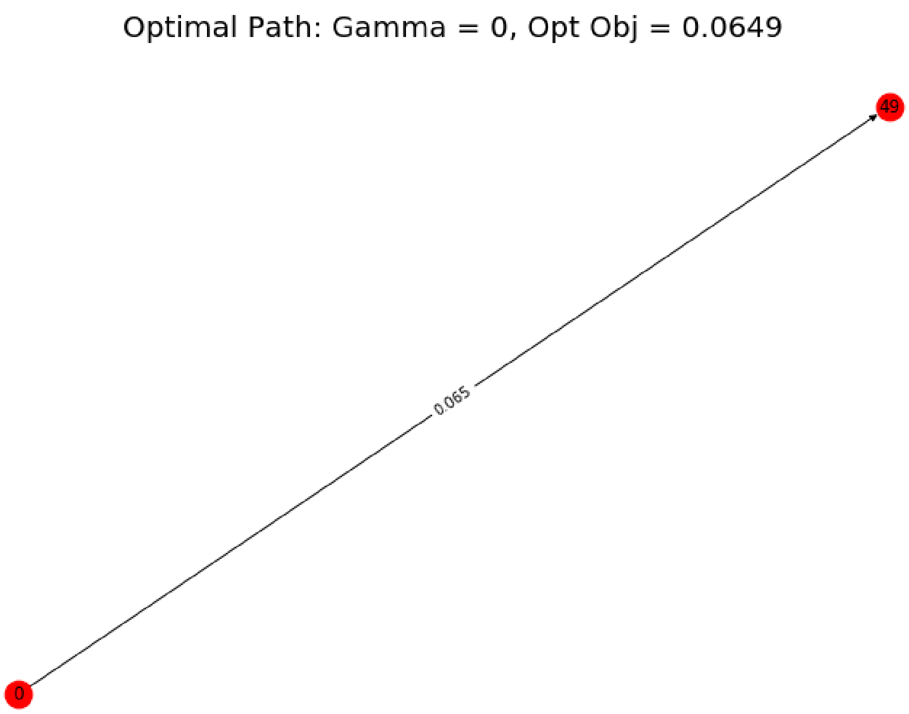
```
Optimize a model with 2261 rows, 4419 columns and 11045 nonzeros
Variable types: 2210 continuous, 2209 integer (2209 binary)
Coefficient statistics:
  Matrix range     [1e-03, 5e+00]
  Objective range  [1e-04, 4e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]

Loaded MIP start with objective 1.33876

Presolve removed 91 rows and 178 columns
Presolve time: 0.01s
Presolved: 2170 rows, 4241 columns, 10600 nonzeros
Variable types: 2121 continuous, 2120 integer (2120 binary)

Root relaxation: objective 1.127922e+00, 65 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    1.12792    0   35    1.33876    1.12792  15.7%     -    0s
     0     0     cutoff    0          1.33876    1.33876  0.00%     -    0s

Cutting planes:
  Gomory: 1
  Clique: 3

Explored 1 nodes (77 simplex iterations) in 0.06 seconds
Thread count was 8 (of 8 available processors)

Solution count 1: 1.33876

Optimal solution found (tolerance 1.00e-04)
Best objective 1.338763884000e+00, best bound 1.338763884000e+00, gap 0.0000%
```

In [11]:
```
# print optimal values and paths, plot network
for o in output:
    print('\nFor Gamma: '+str(o[0]))
    print('Path:')
    print(o[1])
    print('Cost of Movement (Objective):')
    print(o[2])
    networkCompletePlot(o,maxNodes)
    networkPathPlot(o,maxNodes,c)
```

```
For Gamma: 0.0
```

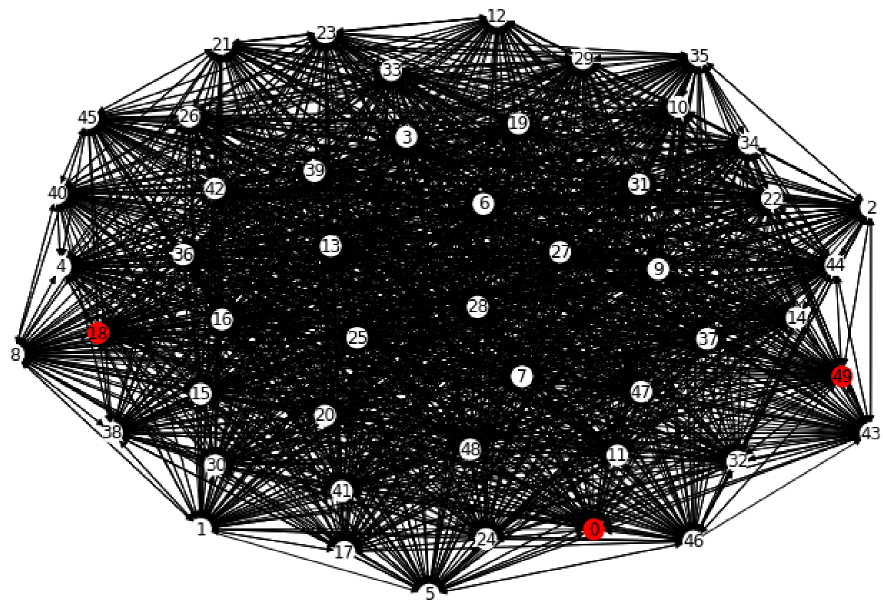
```
Path:
[(0, 49)]
Cost of Movement (Objective):
0.06489829400000001
```



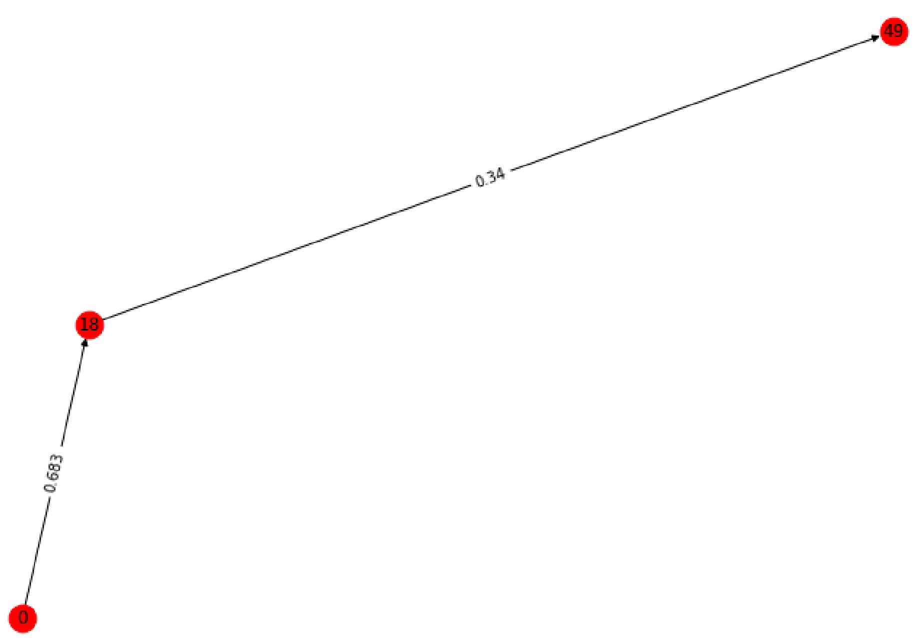



```
For Gamma: 1.0
Path:
[(0, 18), (18, 49)]
Cost of Movement (Objective):
1.2783774319999988
```

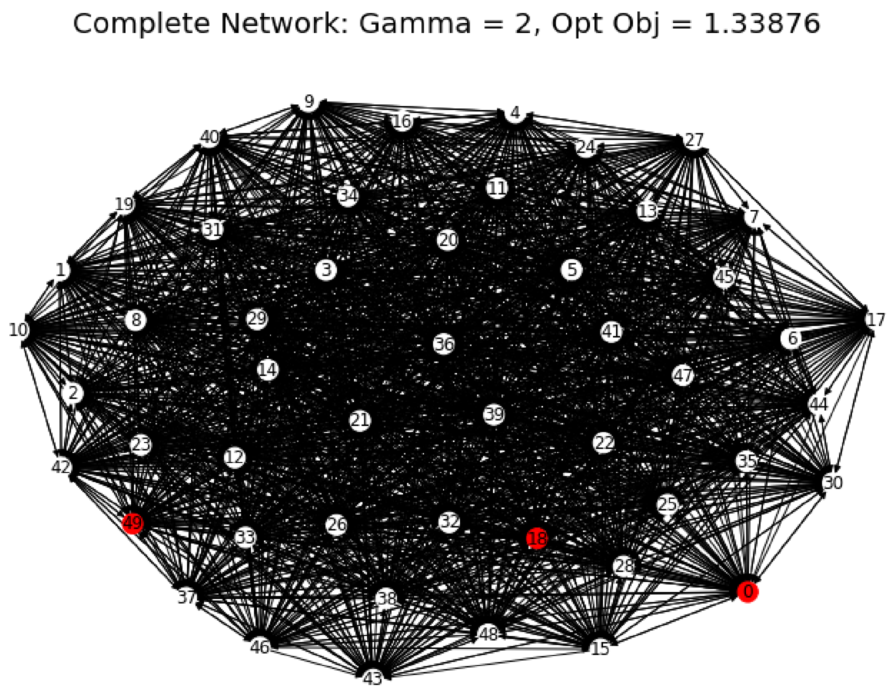
Complete Network: Gamma = 1, Opt Obj = 1.27838



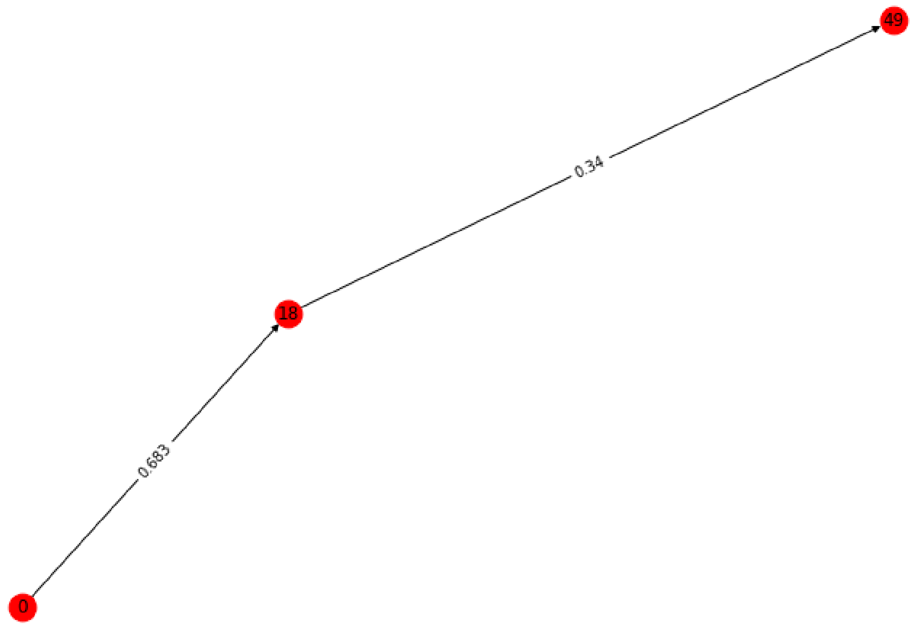
Optimal Path: Gamma = 1, Opt Obj = 1.27838



```
For Gamma: 2.0
Path:
[(0, 18), (18, 49)]
Cost of Movement (Objective):
1.3387638840000002
```

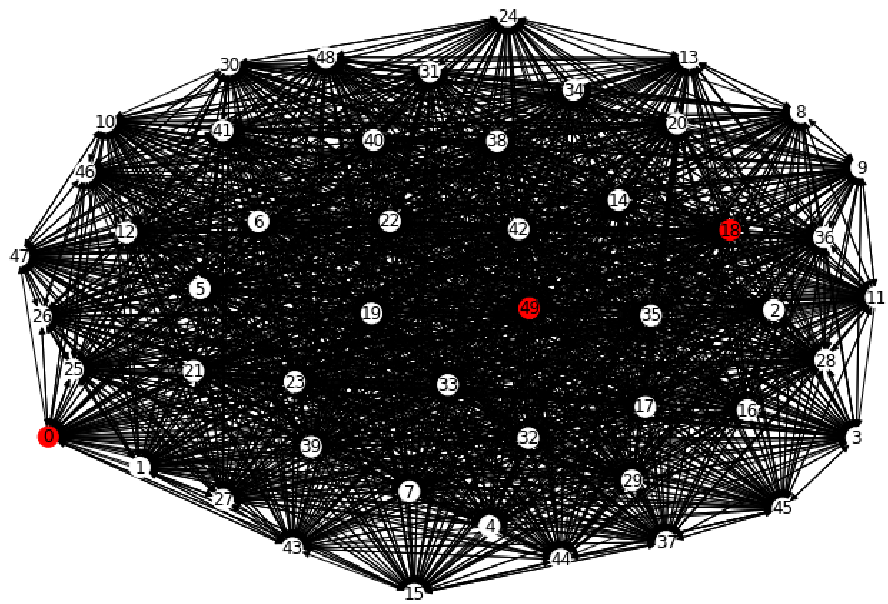


Optimal Path: Gamma = 2, Opt Obj = 1.33876

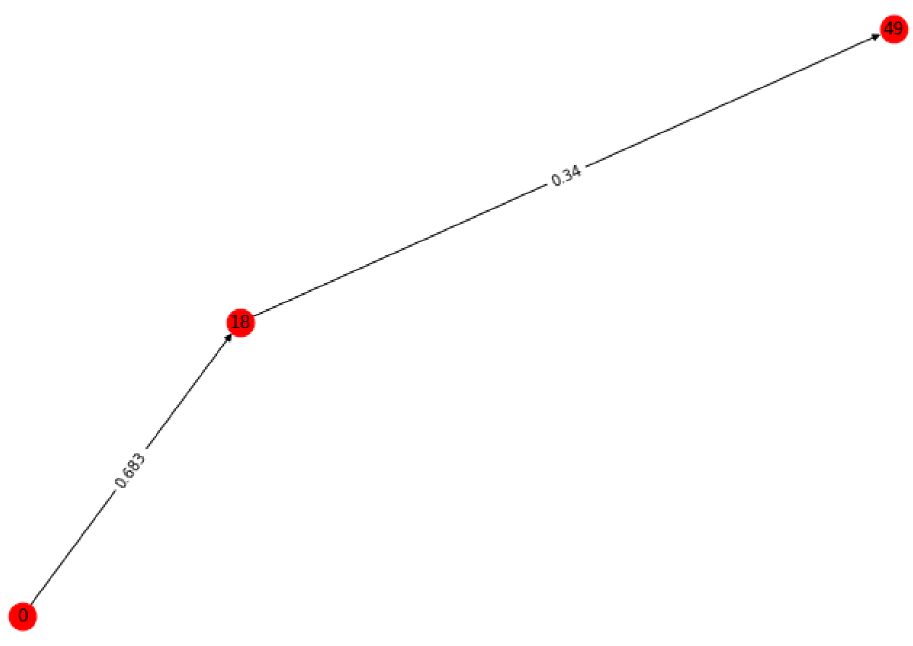



```
For Gamma: 3.0
Path:
[(0, 18), (18, 49)]
Cost of Movement (Objective):
1.338763884
```

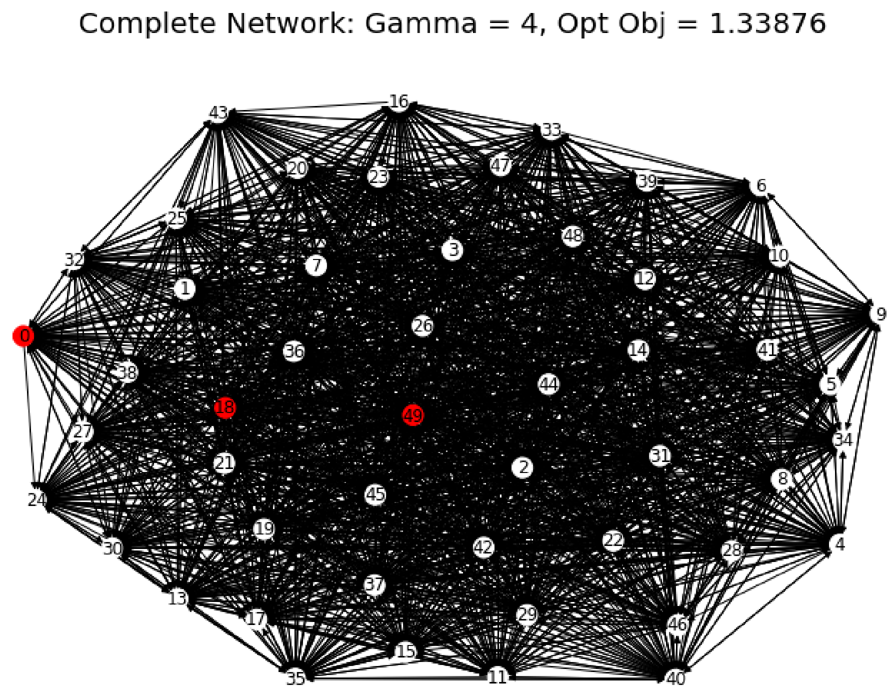
Complete Network: Gamma = 3, Opt Obj = 1.33876



Optimal Path: Gamma = 3, Opt Obj = 1.33876



```
For Gamma: 4.0
Path:
[(0, 18), (18, 49)]
Cost of Movement (Objective):
1.338763884
```



Optimal Path: Gamma = 4, Opt Obj = 1.33876