```python
import numpy as np
import numpy.random as npr
import scipy.stats as stats
import matplotlib.pyplot as plt
import pandas as pd


pi = np.pi

# euclidian distance
def dist(x,y):
    z = np.sqrt(np.sum( (x - y) ** 2))
    return z

def T(k):
    return  1 /  (np.sqrt(1+k))

# Calculate cost of total system
def V(x,P):
    cost = 0
    n = len(x)

    for i in range(0,n-1):
        cost += P[x[i],x[i+1]]

    cost += P[x[n-1],x[0]]
    return cost


# We will debug with points on the unit circle
npoints = 10
Nsamples = 500000

theta = np.array([2*pi*j/npoints for j in range(npoints)])
x = np.cos(theta)
y = np.sin(theta)
p = np.column_stack((x,y))

Pcircle = np.zeros((npoints,npoints))

# Calculate disttance matrix
for i in range(npoints):
    p1 = p[i,:]
    for j in range(npoints):
        p2 = p[j,:]
        Pcircle[i,j] = dist(p1,p2)
```

```
# Save optimal permutation
bestroute = np.array([i for i in range(npoints)])
bestroutecost = V(bestroute,Pcircle)

plt.figure()
plt.plot(x[bestroute],y[bestroute],'.-')
plt.title('Optimal')
plt.show()

# Initialize Metropolis-Hastings and best perm along with best
 cost
perm = npr.permutation(npoints) # bestroute + 0#
bestperm = perm + 0
bestcost = V(bestperm,Pcircle) + 0


plt.figure()
plt.plot(x[perm],y[perm],'.-')
plt.title('Start')
plt.show()

for i in range(Nsamples):
    print(i)

    # Perform proposal jump
    proposal = perm
    p1 = npr.randint(npoints)
    p2 = npr.randint(npoints)
    while p1 == p2:
        p2 = npr.randint(npoints)
    proposal[p1],proposal[p2] = perm[p2],perm[p1]


    # Perform Metropolis-Hastings step - do it like this to
avoid numeric problems
    ratio = np.exp(-(V(proposal,Pcircle) - V(perm,Pcircle))/T(
i))


    if ratio > 1:
        perm = proposal + 0
    else:
        U = npr.rand()
        if ratio > U:
            perm = proposal + 0

    # Calculate cost
    cost = V(perm,Pcircle)

    if cost < bestcost:
```

```
        bestperm = perm + 0
        bestcost = cost + 0

plt.figure()
plt.plot(x[bestperm],y[bestperm],'.-')
plt.title('Optimal from MH')
plt.show()


## Try on real data from inside
df = pd.read_excel (r'cost.xlsx', sheet_name='Sheet1')
P = np.array(df)
npoints = P.shape[0]
Nsamples = 2000000

# Initialize Metropolis-Hastings and best perm along with best
 cost
perm = npr.permutation(npoints)
bestperm = perm + 0
bestcost = V(bestperm,P) + 0

# Cooling scheme
def T(k):
    return  1 /  (np.sqrt(1+k))

for i in range(Nsamples):

    print(i)
    # Perform proposal jump
    proposal = perm
    p1 = npr.randint(npoints)
    p2 = npr.randint(npoints)
    while p1 == p2:
        p2 = npr.randint(npoints)
    proposal[p1],proposal[p2] = perm[p2],perm[p1]

    # Perform Metropolis-Hastings step - do it like this to
avoid numeric problems
    ratio = np.exp(-(V(proposal,P) - V(perm,P))/T(i))

    if ratio > 1:
        perm = proposal + 0
    else:
        U = npr.rand()
        if ratio > U:
            perm = proposal + 0

    # Calculate cost
    cost = V(perm,P)
```

```
    if cost < bestcost:
        bestperm = perm + 0
        bestcost = cost + 0
```