FA_TSP

April 5, 2022

```
[1]: import pandas as pd
     import numpy as np
```

```
[3]: df = pd.read_csv('./adj_mat_kota.csv')
     df.head(10)
```

```
[3]:           0         1         2         3         4         5         6  \
     0  0.000000  0.936638  0.713507  0.194283  0.298506  0.067479  0.765765
     1  0.936638  0.000000  0.278226  0.198572  0.547646  0.445650  0.710273
     2  0.713507  0.278226  0.000000  1.201996  0.358448  1.678937  0.175979
     3  0.194283  0.198572  1.201996  0.000000  0.251722  0.802101  0.230487
     4  0.298506  0.547646  0.358448  0.251722  0.000000  0.424933  2.063560
     5  0.067479  0.445650  1.678937  0.802101  0.424933  0.000000  0.297784
     6  0.765765  0.710273  0.175979  0.230487  2.063560  0.297784  0.000000
     7  0.044104  0.663657  0.262226  0.476379  1.735895  0.415713  0.307613
     8  0.763039  1.396479  1.079456  0.529837  0.547884  2.237543  0.788896
     9  1.045118  0.119447  1.769962  1.031190  1.825648  1.042371  0.454101

               7         8         9
     0  0.044104  0.763039  1.045118
     1  0.663657  1.396479  0.119447
     2  0.262226  1.079456  1.769962
     3  0.476379  0.529837  1.031190
     4  1.735895  0.547884  1.825648
     5  0.415713  2.237543  1.042371
     6  0.307613  0.788896  0.454101
     7  0.000000  3.334264  0.800098
     8  3.334264  0.000000  0.932465
     9  0.800098  0.932465  0.000000
```

```
[12]: def calc_dist(X,adj_mat):
          return sum( map( lambda x,y: adj_mat[x,y] ,X,np.roll(X,-1) ))
```

```
[5]: # Params
     n_kota = len(df.columns)
     n_individu = 15
     a = 1
```

```
b = 10

# Generate individu
gen_individu = lambda n_individu,n_kota,a,b: np.random.uniform(␣
 ↪a,b,(n_individu,n_kota))
kunangs = gen_individu(n_individu,n_kota,a,b)
# kunangs
```

```
[6]: def diskritisasi(kunangs):
         return np.argsort(kunangs)
```

```
[10]: def calculate_fitness(kunangs,df):
          d_kunangs = diskritisasi(kunangs)
          fitness = np.array(  list(map( lambda x: calc_dist( x ,df.values) ,␣
       ↪d_kunangs )) )
          fitness = fitness.reshape( (-1,1) )
          return np.concatenate( ( kunangs ,fitness ) ,axis=1)
```

```
[8]: def sort_individu(kunangs_with_f):
         return kunangs_with_f[kunangs_with_f[:,-1].argsort()]
```

```
[13]: kunangs_w_f = sort_individu(calculate_fitness(kunangs,df))
      kunangs_w_f
```

```
[13]: array([[ 3.40123851,  8.58962347,  3.19738508,  6.97304658,  5.17851877,
               9.56169143,  9.74513299,  2.6551885 ,  7.30302749,  1.95573578,
               5.44991016],
             [ 3.14756446,  8.53300104,  3.67175044,  9.52532886,  6.19246062,
               8.56777321,  9.51658189,  8.75585813,  4.8136258 ,  6.93735446,
               5.87968924],
             [ 2.33694875,  1.8835545 ,  1.33951552,  1.77571315,  3.23798275,
               3.9418094 ,  1.17810787,  1.74926042,  1.30457348,  7.84926744,
               5.96207854],
             [ 6.56579578,  5.88584652,  3.99277919,  8.5189323 ,  5.22382859,
               7.00694891,  9.40670599,  6.92758495,  4.5838408 ,  7.79538665,
               6.0514683 ],
             [ 7.25690265,  3.27672632,  4.2667164 ,  4.31991242,  1.491262  ,
               2.23301182,  6.76126586,  5.37737405,  8.81222047,  6.21631705,
               6.15807135],
             [ 5.4061209 ,  9.9642692 ,  7.47988297,  2.82000865,  9.15882535,
               9.31081144,  1.26228316,  8.4652377 ,  3.59473199,  2.64023987,
               7.07065106],
             [ 7.09409022,  3.81976346,  4.74632074,  3.83305323,  1.25288275,
               9.4506047 ,  7.90901475,  4.8717907 ,  3.51571308,  1.70162628,
               7.34997275],
             [ 7.655206  ,  7.31814534,  4.01441254,  1.62274674,  4.91693197,
               6.09796016,  2.73274393,  2.73734226,  5.36880061,  1.57753238,
               7.40279651],
             [ 3.39427086,  1.04805628,  9.78464488,  7.60643719,  6.38872973,
               8.64442285,  5.65541852,  8.75492911,  2.24770104,  4.49478874,
               7.73228468],
             [ 1.85162426,  9.97096442,  6.28403761,  9.62431412,  3.14155601,
               7.14320614,  5.44650286,  7.1837027 ,  3.40934057,  2.61936334,
               8.08976445],
             [ 3.22049325,  9.83344509,  7.58724766,  9.36676591,  9.60180943,
               3.10939315,  2.72570474,  3.9928033 ,  4.26999896,  4.32068614,
               9.15769342],
             [ 4.73576436,  4.18466397,  3.40317593,  4.98764158,  2.90962991,
               1.90668407,  9.40307861,  5.77561283,  8.92366048,  2.85711603,
               9.53293643],
             [ 9.93354981,  5.72902138,  8.99484466,  4.80759783,  3.17993441,
               6.22495495,  1.21166783,  3.78513386,  5.23730776,  2.4901924 ,
              10.02220021],
             [ 3.03424466,  5.94008004,  8.43611071,  6.83233368,  9.85330569,
               2.45616113,  6.21304091,  3.06788387,  5.33091477,  1.38232881,
              10.21154807],
             [ 4.44839019,  6.15005005,  3.93646728,  4.35815755,  2.46387082,
               8.2294093 ,  5.92888312,  6.25094929,  7.05502558,  1.38075324,
              12.33424769]])
```

[14]:

```
def solusi(kunangs_w_f):
    df_kota = pd.DataFrame(diskritisasi(kunangs_w_f[:,:-1]))
    cols = [ 'Urutan ' + str(i+1) for i in range( df_kota.shape[1]) ]
    df_kota.columns = cols
    df_kota['Jarak'] =  kunangs_w_f[:,-1].reshape(-1,1)
    return df_kota
```

[15]:

```
solusi(kunangs_w_f)
```

[15]:

```
    Urutan 1  Urutan 2  Urutan 3  Urutan 4  Urutan 5  Urutan 6  Urutan 7  \
0          9         7         2         0         4         3         8
1          0         2         8         4         9         1         5
2          6         8         2         7         3         1         0
3          2         8         4         1         0         7         5
4          4         5         1         2         3         7         9
5          6         9         3         8         0         2         7
6          4         9         8         1         3         2         7
7          9         3         6         7         2         4         8
8          1         8         0         9         6         4         3
9          0         9         4         8         6         2         5
10         6         5         0         7         8         9         2
11         5         9         4         2         1         0         3
12         6         9         4         7         3         8         1
13         9         5         0         7         8         1         6
```

```
14          9          4          2          3          0          6          1

    Urutan 8  Urutan 9  Urutan 10      Jarak
0          1         5          6   5.449910
1          7         6          3   5.879689
2          4         5          9   5.962079
3          9         3          6   6.051468
4          6         0          8   6.158071
5          4         5          1   7.070651
6          0         6          5   7.349973
7          5         1          0   7.402797
8          5         7          2   7.732285
9          7         3          1   8.089764
10         3         4          1   9.157693
11         7         8          6   9.532936
12         5         2          0  10.022200
13         3         2          4  10.211548
14         7         8          5  12.334248
```

[17]:

```
def movement( X, i,j ,p):
    r = np.linalg.norm(X[i,:] - X[j,:])
    term1 = p['beta0'] * np.exp(-1 * p['gamma'] * r**2  )
    term2 = X[j,:] - X[i,:]
    term3 = p['alpha'] * np.random.uniform(0,1)
    return  X[i,:] +  (  term1 * term2 ) + term3
```

[29]:

```
# Params
p = { 'beta0' : 1, 'gamma': 100 , 'alpha': 3}
n_iter = 100
generasi = 0
n_kota = len(df.columns)
n_individu = 100
a = 1
b = 10

# Inisialisasi
kunangs = gen_individu(n_individu,n_kota,a,b)
kunangs_w_f = sort_individu(calculate_fitness(kunangs,df))

kunangs = kunangs_w_f[:,:-1]
new_kunangs_w_f = np.copy(kunangs_w_f)
new_kunangs = np.copy(kunangs)

# Main FA Program
while generasi<n_iter:

    temp = new_kunangs.copy()
    for i in range(n_individu):
        for j in range(n_individu):
            if ( new_kunangs_w_f[i,-1] > new_kunangs_w_f[j,-1] ) and (i!=j) :
                temp[i,:] = movement(new_kunangs,i,j,p)

    new_kunangs = temp.copy()
    new_kunangs_w_f = sort_individu(calculate_fitness(new_kunangs,df))
#     print('iterasi: ',generasi)
#     print(solusi(new_kunangs_w_f).head())
#     print(new_kunangs_w_f)
#     print('======================')
    generasi = generasi+1


# Print Fireflies
solusi(new_kunangs_w_f)
```

[29]:

```
    Urutan 1  Urutan 2  Urutan 3  Urutan 4  Urutan 5  Urutan 6  Urutan 7  \
0          7         2         6         8         4         5         3
1          6         2         5         1         9         8         3
2          0         4         9         6         8         3         1
3          1         9         7         2         6         5         3
4          4         5         1         0         2         7         9
..       …       …       …       …       …       …       …
95         0         7         8         1         4         6         9
96         7         8         0         6         1         9         3
97         9         2         7         8         0         5         3
98         0         1         6         4         8         7         2
99         5         2         4         6         1         8         7

    Urutan 8  Urutan 9  Urutan 10      Jarak
0          1         9          0   4.409259
1          4         0          7   4.784259
2          2         7          5   5.119204
3          4         0          8   5.167379
4          6         3          8   5.345362
..       …       …         …         …
95         2         3          5  11.681691
96         2         5          4  11.765739
97         1         6          4  11.797123
98         3         5          9  11.946430
99         9         0          3  12.383561

[100 rows x 11 columns]
```

[30]:

```
new_kunangs_w_f
```

```
[30]: array([[  9.92721079,   8.82377011,   3.12220798, …,   5.10062378,
                 9.09060472,   4.40925928],
              [155.56079942, 152.28925038, 150.03252448, …, 152.65418329,
               152.34976645,   4.78425915],
              [127.30294967, 134.03509468, 134.74097543, …, 133.21479406,
               131.52736341,   5.11920354],
              …,
              [154.09666212, 156.41271977, 151.0932861 , …, 153.83649652,
               151.05798933,  11.79712262],
              [150.54258354, 150.56029135, 151.91975814, …, 151.7249015 ,
               158.50672665,  11.94642977],
              [162.41119238, 157.53951259, 154.58005982, …, 158.36585286,
               160.38364143,  12.38356122]])
```