CS_TSP

April 26, 2022

[11]:
```
import pandas as pd
import numpy as np
from scipy.stats import levy_stable as l
```

[12]:
```
df = pd.read_csv('./adj_mat_kota.csv')
df.head(10)
```

[12]:
```
          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
```

[13]:
```
def calc_dist(X,adj_mat):
    return sum( map( lambda x,y: adj_mat[x,y] ,X,np.roll(X,-1) ))
```

[14]:
```
# 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))
cuckoos = gen_individu(n_individu,n_kota,a,b)
# Cuckoos adalah representasi dari telur cuckoo yang disimpan dalam sarang.
# Asumsi 1 sarang hanya terdiri dari 1 telur cuckoo
# Tiap generasi akan membawa sarang terbaik dimana fitness individu cuckoo␣
↪terbaik
```

[15]:
```
def diskritisasi(cuckoos):
    return np.argsort(cuckoos)
```

[16]:
```
def calculate_fitness(cuckoos,df):
    d_cuckoos = diskritisasi(cuckoos)
    fitness = np.array(  list(map( lambda x: calc_dist( x ,df.values) ,␣
↪d_cuckoos )) )
    fitness = fitness.reshape( (-1,1) )
    return np.concatenate( ( cuckoos ,fitness ) ,axis=1)
```

[17]:
```
def sort_individu(cuckoos_with_f):
    return cuckoos_with_f[cuckoos_with_f[:,-1].argsort()]
```

[18]:
```
cuckoos_w_f = sort_individu(calculate_fitness(cuckoos,df))
# cuckoos_w_f
```

[19]:
```
def solusi(cuckoos_w_f):
    df_kota = pd.DataFrame(diskritisasi(cuckoos_w_f[:,:-1]))
    cols = [ 'Urutan ' + str(i+1) for i in range( df_kota.shape[1]) ]
    df_kota.columns = cols
    df_kota['Jarak'] =  cuckoos_w_f[:,-1].reshape(-1,1)
    return df_kota
```

[20]:
```
solusi(cuckoos_w_f)
```

[20]:
```
   Urutan 1  Urutan 2  Urutan 3  Urutan 4  Urutan 5  Urutan 6  Urutan 7  \
0         1         9         7         5         4         3         6
1         2         7         9         5         6         8         3
2         3         2         6         7         9         0         5
3         2         7         9         5         3         8         4
4         4         8         3         0         9         6         1
5         3         2         4         1         5         9         8
6         4         1         9         7         0         3         6
7         6         3         4         7         1         9         0
8         2         1         7         8         4         0         3
```

```
9          3         8         5         7         9         0         4
10         6         5         0         4         9         2         1
11         6         2         9         0         1         3         4
12         9         3         1         7         4         6         5
13         8         5         7         9         2         4         6
14         2         9         8         7         1         4         6

    Urutan 8  Urutan 9  Urutan 10      Jarak
0          8         0          2   4.786067
1          1         0          4   5.513375
2          1         4          8   5.669299
3          1         0          6   6.410545
4          7         5          2   6.598252
5          6         0          7   6.603720
6          8         5          2   6.999889
7          5         8          2   7.606783
8          5         9          6   7.791372
9          6         2          1   8.043151
10         3         7          8   9.335715
11         5         7          8   9.341796
12         2         0          8  10.078607
13         0         3          1  10.200424
14         3         0          5  11.482741
```

[21]:
```python
def movement( X , p):
    levy = l.rvs( p['lb'][0] , p['lb'][1] , size=X.shape)
    return  X + p['alpha'] * levy
```

[22]:
```python
def selec(new_cuckoos):
    pa = np.round( np.random.uniform(0,1) * (new_cuckoos.shape[0] -1) ).
 ↪astype(int)
    sz = new_cuckoos[:-pa][:].shape
    new_cuckoos[:-pa,:] = np.random.uniform(a,b, size=(sz))
    return new_cuckoos
```

[28]:
```python
# Inisialisasi Parameter
n_iter = 100
generasi = 0
n_kota = len(df.columns)
n_individu = 100

p = { 'alpha': 1, 'lb':[1.8,-0.5] }
a = 1
b = 10

# Inisialisasi - Generasi Pertama
cuckoos = gen_individu(n_individu,n_kota,a,b)
```

```
cuckoos_w_f = sort_individu(calculate_fitness(cuckoos,df))
cuckoos = cuckoos_w_f[:,:-1]
new_cuckoos_w_f = np.copy(cuckoos_w_f)
new_cuckoos = np.copy(cuckoos)

# Main Program
while generasi<n_iter:
    #bangkitkan cuckoo secara acak dengan levy flight
    new_cuckoos = movement(new_cuckoos,p)

    #evaluasi fitness cuckoo
    new_cuckoos_w_f = sort_individu(calculate_fitness(new_cuckoos,df))

    # seleksi
    new_cuckoos = selec(new_cuckoos)



    #next generasi
    generasi = generasi+1


# Print Best
solusi(new_cuckoos_w_f)
```

[28]:

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

    Urutan 8  Urutan 9  Urutan 10      Jarak
0          7         5          3   4.089377
1          2         0          3   4.440742
2          5         1          7   4.476509
3          7         9          1   4.793965
4          3         9          8   4.940298
..       …       …        …        …
95         6         3          7  11.404114
96         3         9          1  11.783153
```

```
97          1          8          7  12.185274
98          0          9          2  12.880662
99          7          8          5  13.856338

[100 rows x 11 columns]
```