TSP_ABC

April 27, 2022

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

df = pd.read_csv('./adj_mat_kota.csv')
```

[2]:
```
gen_individu = lambda n_individu,n_kota,a,b: np.random.uniform(␣
↪a,b,(n_individu,n_kota))

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

def diskritisasi(bees):
    return np.argsort(bees)

def calculate_fitness(bees,df):
    d_bees = diskritisasi(bees)
    fitness = np.array(  list(map( lambda x: calc_dist( x ,df.values) , d_bees␣
↪)) )
    fitness = fitness.reshape( (-1,1) )
    return fitness

def sort_individu(fitness):
    return np.argsort(fitness)


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

def generate_tipe_bee(presentase,bees):
    proporsi = np.array(presentase) * bees.shape[0]
    proporsi[-1] = bees.shape[0] - ( np.sum(proporsi) - proporsi[-1] )
    return np.concatenate( [ np.repeat( i , round(p)  ) for i,p in␣
↪enumerate(proporsi)  ] )
```

```
def scout_movement(scout,a,b):
    return scout + np.random.uniform(a,b,size=scout.shape)

def employed_movement(employed,alpha):
    return employed + np.random.uniform(0,1,size=employed.shape) * alpha

def waggle_dance(bees,tipe,fitness):
    df = pd.DataFrame( np.concatenate( (bees,tipe.
 ↪reshape((-1,1)),fitness),axis=1 ) )
    employed = df[ df.iloc[:,-2] == 0 ]
    p = employed.iloc[:,-1]
    if p.sum() == 0:
        p = p + 1
    return employed.sample(n=1,weights=p).iloc[:,:-2].values

def onlooker_movement(onlooker,beta,bees,tipe,fitness):
    term1 = np.random.uniform() * ( onlooker - waggle_dance(bees,tipe,fitness) )
    term2 = beta * np.random.uniform(size=onlooker.shape)
    return onlooker + term1  + term2

def get_bee_by_type(df,x):
    return df[df[df.columns[-2]] == x].iloc[:,:-2].values

def movement(bees,tipe,fitness,params):
    df = pd.DataFrame(np.concatenate((bees,tipe.
 ↪reshape((-1,1)),fitness),axis=1))

    employed = employed_movement( get_bee_by_type(df,0), params['alpha'])
    onlooker =  onlooker_movement( get_bee_by_type(df,1),params['beta'],␣
 ↪bees,tipe,fitness )
    scouts = scout_movement( get_bee_by_type(df,2),params['a'] , params['b'] )
    new_bee = np.concatenate( (employed,onlooker,scouts) )

    return new_bee

def seleksi(bees,tipe,fitness,params):
    idxs = sort_individu(fitness.flatten())
    return bees[idxs] , tipe , fitness[idxs]


def inisialisasi(params,df):
    return␣
 ↪gen_individu(int(params['n_individu']),int(params['n_kota']),params['a'],params['b'])

def ABC(params,df):
    generasi = 0
    bees = inisialisasi(params,df)
    tipe = generate_tipe_bee(params['presentase'],bees)
    fitness = calculate_fitness(bees,df)

    while generasi<params['max_generasi']:

        bees = movement(bees,tipe,fitness,params)
        fitness = calculate_fitness(bees,df)
        bees , tipe , fitness = seleksi(bees,tipe,fitness,params)

        generasi = generasi+1

    return solusi(np.concatenate((bees,fitness),axis=1))

def run_BCO(dfparams,df):
    return [ ABC( dfparams.loc[i].to_dict() ,df) for i in range( dfparams.
  ↪shape[0]) ]

def save_BCO(hasils):
    for h in enumerate(hasils):
        pd.DataFrame(h[1]).to_csv('hasil/hasil_' + str(h[0]) + '.csv')
```

[3]:
```
# TIPE BEE
# 0 = Employed , 1 = Onlooker , 2 = Scout
params = {
    "n_individu":10,
    "n_kota":10,
    "a":-1,
    "b":1,
    "alpha": 1, # ukuran exploitasi employed
    "beta":1, # kecepatan onlooker mendekati employed
    "max_generasi":100,
    "presentase": [0.20,0.25,0.55]

}

ABC(params,df)
```

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

```
6          6          2          9          0          5          7          1
7          4          5          1          0          3          7          2
8          2          0          1          5          7          9          4
9          2          3          1          8          9          4          6

   Urutan 8  Urutan 9  Urutan 10      Jarak
0         5         0          6   4.113384
1         5         0          6   4.113384
2         4         8          3   4.113384
3         7         0          6   4.594528
4         5         0          6   5.595324
5         3         2          8   6.363059
6         8         4          3   6.564480
7         9         8          6   8.294992
8         6         8          3   9.721542
9         0         7          5  10.523240
```

[8]:
```python
# Main Program
dfparams = pd.read_csv('./params/FA_params.csv')
dfparams['n_kota'] = 10
```

[9]:
```python
hasils = run_FA(dfparams,df)
```

[14]:
```python
# hasils[1]
```

[15]:
```python
save_FA(hasils)
```