TSP ACO

April 28, 2022

[1]:

```python
import numpy as np
import pandas as pd

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

[20]:

```python
def generate_1_semut(ndim):
    return [ [], list(range(ndim)) ]

def generate_individus(n_individu,ndim):
    return [ generate_1_semut(ndim) for i in range(n_individu) ]

def generate_pheromones(ndim,tau_awal):
    pheromones = np.ones(shape=(ndim,ndim)) * tau_awal
    np.fill_diagonal(pheromones,0)
    return pd.DataFrame(pheromones)

def generate_delta_pheromones(ndim):
    return generate_pheromones(ndim,0)

def f(X,adj_mat,tipe='full'):
    return sum([  adj_mat[ X[i],X[i+1] ] for i in range(len(X) - 1) ]) + ( adj_mat[ X[-1],X[0] ] if tipe=='full' else 0)

def ro_1(el_c,el_t,pher,adj_mat,params):
    return ( pher[el_c,el_t] )**params['alpha'] * ( 1/adj_mat[el_c,el_t] )**params['beta']

def ro_1_all(semut,pher,adj_mat,params):
    return [  ro_1(semut[0][-1],el_t,pher,adj_mat,params) for el_t in semut[1] ]

def transition_prob(semut,pher,adj_mat,params):
    r = ro_1_all(semut,pher,adj_mat,params)
    total_prob = sum(r)
    return [ p/total_prob for p in r ]

def choose_next_kota(semut,pher,adj_mat,params):
    df = pd.DataFrame( {'next':semut[1] , 'prob':
 ↪transition_prob(semut,pher,adj_mat,params) } )
    choosen = df.sample(n=1,weights='prob').iloc[0,0]
    return nv_to_v(semut,choosen)

def choose_next_kota_all(semuts,pher,adj_mat,params):
    return [ choose_next_kota(s,pher,adj_mat,params) for s in semuts ]

def update_delta_pher(semuts,delta_pher,adj_mat,params):
    temp = generate_delta_pheromones(params['ndim'])
    for s in semuts:
        Lk = f(s[0],adj_mat,tipe='partial')
        temp[ s[0][-2] ][ s[0][-1] ] = delta_pher[ s[0][-2] ][ s[0][-1] ] +␣
 ↪(params['Q']/Lk)
    return temp

def update_pheromones(pher,delta_pher,semuts,adj_mat,params):
    return params['rho'] * pher +␣
 ↪update_delta_pher(semuts,delta_pher,adj_mat,params)

def sol(semuts):
    return np.array( [ s[0] for s in semuts])

def calc_fitness(semuts,adj_mat):
    return np.array( [ f(s,adj_mat,'full') for s in semuts ] )

def find_best(sl,fitness):
    idxs = np.argsort(fitness)
    return np.append( sl[idxs][0],fitness[idxs][0]   )

def satu_full(pheromones,adj_mat,params):

    semuts = generate_individus(params['n_individu'],params['ndim'])
    semuts = inisialisasi(semuts)
    delta_pheromones = generate_delta_pheromones(params['ndim'])

    t = 1
    while t < params['ndim']:
        semuts = choose_next_kota_all(semuts,pheromones.values,adj_mat,params)
        pheromones =␣
 ↪update_pheromones(pheromones,delta_pheromones,semuts,adj_mat,params)
        delta_pheromones = generate_delta_pheromones(params['ndim'])
        t = t + 1

    sl = sol(semuts)
    fitness = calc_fitness(sl,adj_mat)
    return find_best(sl,fitness) , pheromones
```

```
def ACO(params,df):
    adj_mat = df.values
    generasi = 0
    bests = []
    pheromones = generate_pheromones(params['ndim'],2)
    delta_pheromones = generate_delta_pheromones(params['ndim'])

    while generasi < params['max_generasi']:
        best_ , pheromones = satu_full(pheromones,adj_mat,params)
        bests.append(best_)
        generasi = generasi + 1

    return np.array(bests)

def nv_to_v(semut,el):
    temp = semut.copy()
    temp[0].append(el)
    temp[1].remove(el)
    return temp

def inisialisasi(semuts):
    return [ nv_to_v(s,s[1][ np.random.randint(0,len(s[1]))] ) for s in semuts ]
```

[23]:

```
params = {
    'alpha':1,
    'beta':1,
    'rho':0.9,
    'Q':1,
    'n_individu':10,
    'ndim':10,
    'max_generasi':5
}

hasil = ACO(params,df)
df_hasil = pd.DataFrame(hasil)
```

[24]:

```
df_hasil
```

[24]:

```
     0    1    2    3    4    5    6    7    8    9         10
0  4.0  8.0  9.0  1.0  3.0  6.0  2.0  7.0  0.0  5.0  3.003576
1  0.0  5.0  6.0  2.0  1.0  9.0  8.0  4.0  3.0  7.0  3.191468
2  6.0  2.0  4.0  3.0  1.0  9.0  8.0  0.0  7.0  5.0  3.557272
3  2.0  6.0  5.0  0.0  7.0  3.0  1.0  9.0  8.0  4.0  3.218540
4  3.0  4.0  2.0  7.0  0.0  5.0  6.0  8.0  9.0  1.0  3.321142
```