TSP_PSO

April 26, 2022

[6]:
```
import numpy as np
import pandas as pd

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

[80]:
```
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(partikels):
    return np.argsort(partikels)

def calculate_fitness(partikels,df):
    d_partikels = diskritisasi(partikels)
    fitness = np.array(  list(map( lambda x: calc_dist( x ,df.values) ,␣
 ↪d_partikels )) )
    return fitness

def idx_sort_individu(fitnesses):
    return fitnesses.argsort()

def sort_PX(P,X,fts):
    return P[fts] , X[fts]

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

def new_v( V, X , P, G ,p):
    r1 = np.random.uniform(0,1)
    r2 = np.random.uniform(0,1)
    term1 = p['W1'] * r1 * ( P-X )
```

```
    term2 = p['W2'] * r2 * ( G-X)
    return  V + term1 + term2

def new_v_all(V,X,P,p):
    return np.array( [ new_v(V[i],X[i],P[i],P[0],p) for i in range(V.shape[0])
 ↪] )

def new_x_all(X,V):
    return X + V

def new_p(P,X_new):
    return P if P[-1] > X_new[-1] else X_new

def new_p_all(P,X,fts_P,fts_X):
    return np.array( [ P[i,:] if fts_P[i] > fts_X[i] else X[i,:] for i in
 ↪range(P.shape[0]) ] )


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

def inisialisasi_v(params,X):
    return np.zeros_like(X)

def inisialisasi_p(params,X):
    return X.copy()

def PSO(params,df):

    generasi = 0

    X = inisialisasi(params,df)
    V = inisialisasi_v(params,X)
    P = inisialisasi_p(params,X)

    fts_P = calculate_fitness(P,df)
    idxs_1 = idx_sort_individu(fts_P)
    P,V = sort_PX(P,X,idxs_1)


    while generasi<params['max_generasi']:

        V = new_v_all(V,X,P,params)
        X = new_x_all(X,V)
```

```
        fts_X = calculate_fitness(X,df)
        idxs_2 = idx_sort_individu(fts_X)
        P,X = sort_PX(P,X,idxs_2)

        P = new_p_all(P,X,fts_P[idxs_2],fts_X[idxs_2])
        fts_P = calculate_fitness(P,df)
        idxs_1 = idx_sort_individu(fts_P)
        P,X = sort_PX(P,X,idxs_1)

        generasi = generasi+1


    return solusi(np.concatenate((P,fts_P[idxs_1].reshape(-1,1)), axis=1))

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

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

[97]:
```
params = {
    "n_individu":500,
    "n_kota":10,
    "a":-4,
    "b":4,
    "W1":2,
    "W2":2,
    "max_generasi":100
}

PSO(params,df)
```

[97]:

	Urutan 1	Urutan 2	Urutan 3	Urutan 4	Urutan 5	Urutan 6	Urutan 7	\
0	2	7	6	8	3	9	1	
1	0	8	6	9	1	3	2	
2	0	8	9	6	2	1	7	
3	5	0	4	8	3	6	1	
4	4	5	7	2	3	6	9	
..	…	…	…	…	…	…	…	
495	7	4	9	2	3	1	0	
496	9	1	6	4	7	8	5	
497	7	3	2	9	4	6	1	
498	6	7	8	5	2	3	1	
499	4	9	0	1	3	2	5	

```
     Urutan 8  Urutan 9  Urutan 10      Jarak
0           5         0          4   4.209290
1           7         5          4   4.927428
2           5         3          4   5.035508
3           9         7          2   5.245174
4           1         8          0   5.566927
..        …         …        …         …
495         6         5          8  14.304066
496         2         3          0  14.321316
497         0         5          8  14.623740
498         0         9          4  14.829889
499         8         7          6  14.829889

[500 rows x 11 columns]
```

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

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

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

[15]:
```
save_PSO(hasils)
```