KP_ABC

April 27, 2022

[4]:

```
import numpy as np
import pandas as pd

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

[16]:

```
gen_individu = lambda n_individu,n_barang,a,b: np.random.uniform(␣
 ↪a,b,(n_individu,n_barang))

def f_constrain(X,df,lim):
    return np.sum( X* df['Weight'].values ) <= lim

def f_profit(X,df):
    return np.sum(X * df['Profit'].values)

def f_obj(X,df,lim):
    return f_profit(X,df) if f_constrain(X,df,lim) else 0

def diskritisasi(bees):
    return np.round( 1/ ( 1 + np.exp(-1 * bees) ) )

def calculate_fitness(bees,df,p):
    d_bees = diskritisasi(bees)
    fitness = np.array(  list(map( lambda x:f_obj(x,df,p['lim']) , d_bees )) )
    fitness = fitness.reshape( (-1,1) )
    return fitness

def sort_individu(fitness):
    return np.argsort(fitness)[::-1] #bees_with_f[bees_with_f[:,-1].argsort()[::
 ↪-1]]

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

```
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_barang']),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,params)

    while generasi<params['max_generasi']:

        bees = movement(bees,tipe,fitness,params)
        fitness = calculate_fitness(bees,df,params)
        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')
```

[19]:

```
# TIPE BEE
# 0 = Employed , 1 = Onlooker , 2 = Scout
params = {
    "n_individu":100,
    "n_barang":10,
    "a":-4,
    "b":4,
    "alpha": 1, # ukuran exploitasi employed
    "beta":1, # kecepatan onlooker mendekati employed
    "max_generasi":100,
    "presentase": [0.20,0.25,0.55],
    "lim":30

}

ABC(params,df)
```

```
/tmp/ipykernel_39279/614318511.py:13: RuntimeWarning: overflow encountered in
```

```
exp
  return np.round( 1/ ( 1 + np.exp(-1 * bees) ) )
```

[19]:

```
    Barang 1  Barang 2  Barang 3  Barang 4  Barang 5  Barang 6  Barang 7  \
0        1.0       1.0       0.0       1.0       1.0       1.0       0.0
1        1.0       1.0       0.0       1.0       1.0       1.0       0.0
2        1.0       1.0       0.0       1.0       1.0       1.0       0.0
3        1.0       1.0       0.0       1.0       1.0       1.0       0.0
4        0.0       1.0       1.0       1.0       0.0       1.0       0.0
..         …       …       …       …       …       …       …
95       0.0       0.0       1.0       1.0       0.0       1.0       1.0
96       0.0       0.0       1.0       1.0       0.0       1.0       1.0
97       1.0       1.0       1.0       1.0       1.0       1.0       1.0
98       0.0       0.0       1.0       1.0       0.0       1.0       1.0
99       1.0       0.0       1.0       1.0       1.0       0.0       1.0

    Barang 8  Barang 9  Barang 10  Profit
0        1.0       0.0        1.0   370.0
1        1.0       0.0        1.0   370.0
2        1.0       0.0        1.0   370.0
3        1.0       0.0        1.0   370.0
4        0.0       1.0        1.0   360.0
..         …       …         …     …
95       0.0       1.0        1.0     0.0
96       0.0       1.0        1.0     0.0
97       1.0       0.0        0.0     0.0
98       0.0       1.0        1.0     0.0
99       1.0       0.0        1.0     0.0

[100 rows x 11 columns]
```

[ ]:

```
# Main Program
dfparams = pd.read_csv('./params/FA_params_KP.csv')
dfparams['n_barang'] = 10
```

[ ]:

```
hasils = run_FA(dfparams,df)
```

[ ]:

```
# hasils[1]
```

[ ]:

```
save_FA(hasils)
```