```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 30 11:19:21 2019

@author: hernan
EA Seminar
LIA - MODO International Lab
EA Seminar
Version 1.0 31-07-2019

Nsgaii, multi objective evolutionary algorithm
Deb et. al.

"""
import random, copy
import ea_base as ea
import moea_base as moea


def nsgaii_survival_selection(pop, popsize):
    """Nsgaii's survival selection method
    Reduces pop to size popsize

    Parameters
    ----------
    pop: Population
        Population to truncate
    psize: integer
        Population size to which pop is reduced


    Comment
    -------
    The reduced population remains in pop

    """

    """ get number of objectives from the first individual
    in the population """
    nobj = pop[0].nobj
    """ non-dominated front sorting
        fronts contains a list of all fronts
        pop is empty after this
    """
    fronts = moea.non_dominated_sorting(pop)
    #print(len(pop))

    """ copy fronts while there are available spots in pop  """
    i = 0
    while len(pop) < popsize and len(pop) + len(fronts[0]) <= popsize:
        """ use list's pop(0) method  to get front at top of the list ""
        fi = fronts.pop(0)
        moea.front_rank(fi, i)
        moea.crowding_distance(fi, nobj)
        """ extend() method adds the specified list elements
            to the end of the current list
        """
        pop.extend(fi)
 #       print("--- front ", i)
 #       fi.printpop()
        i += 1

    """ if there are still empty spots in pop
        sort the next front by crowding distance
        and get the requiered number to fill pop
    """
    if len(pop) < popsize:
        fi = fronts.pop(0)
        moea.front_rank(fi, i)
        moea.crowding_distance(fi, nobj)
        fi.sort(key = lambda ind: ind.rank[1], reverse = True)
        pop.extend(fi[: popsize - len(pop)])
        print("--- front does not fit")
        #fi.printpop()
#        print("--- front ", i)
#        fi.printpop()

    print("--- pop ")
    pop.printpop()

    if len(pop) != popsize:
        print("Somethig is wrong, len(pop) = " +
              str(len(pop)) + ", popsize = " + str(popsize))



def nsgaii(evaluate=None, select=None, recombine=None, mutate=None,
    seed=None, psize=None, nobj=None, nvar=None, vlow=None, vhigh=None,
    initype=None, ngen=None, pcx=None, pmut=None, keepclones = False):

    """ Nsgaii algorithm, Deb et. al.
    Includes the option to delete clones

    Parameters
    ----------
    evaluate: function name
        Fitness function
    select: function name
        Parent selection method
```

```
recombine: function name
    Recombination method
mutate: function name
    Mutation method
seed: integer
    Seed for the random generator
psize: integer
    Population size
nvar: integer
    Number of variables
vlow: integer (float)
    Lover bound for variables
vhigh : integer (float)
    Upper bound for variables (if binary representation,
    initial probability of ones)
initype: string
    Individual representation. So far: 'binary'
ngen: integer
    Number of generations
pcx: real number in [0.0, 1.0]
    Probability of crossover between two individuals
pmut: real number in [0.0,1.0]
    Probability of mutation per variable
keepclones: boolean
    Instruction to keep clones in the population if set to True.
    Default is False

Returns
-------
Population
    The evolved population

"""

""" Set random generator """
random.seed(seed)

""" Initial population  """
pop = ea.Population(size=psize, nobj=nobj, nvar=nvar, vlow=vlow,
                    vhigh=vhigh, initype=initype)

""" Evaluate the initial population """
for ind in pop:
    ind.fitness = evaluate(ind)

""" Output the population """
f = open("pop_g0.txt", "w")
nsgaii_survival_selection(pop, psize)
pop.fprintpop(f)
f.close()
```

```
print(' --Generation 0')

for g in range(1, ngen+1):
    """ Select the next generation individuals """
    parents = select(pop, psize)
    """ Clone the selected individuals """
    offspring = copy.deepcopy(parents)

    """ Apply crossover and mutation on the offspring """
    for ind1, ind2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < pcx:
            recombine(ind1, ind2)
        mutate(ind1, pmut)
        mutate(ind2, pmut)

    """ Evaluate offspring population """
    for ind in offspring:
        ind.fitness = evaluate(ind)

    """ Delete clones """
    if keepclones == False:
        join_pop = []
        for ind in (pop+offspring):
            if ind not in join_pop:
                join_pop.append(ind)
        pop[:] = join_pop[:]
    else:
        pop.extend(offspring)

    """ Truncate the population (extinctive selection)
        The new population is the best among pop and offspring
    """
    nsgaii_survival_selection(pop, psize)

    """ Output the population """
    f = open("pop_g" + str(g) + ".txt", "w")
    pop.fprintpop(f)
    f.close()

    if g%(ngen/4) == 0:
        print(' --Generation ', g)

print('Ends Nsgaii')
return pop
```