**Deap: 遺傳演算法演算法解決 背包問題**

原創 2017年10月16日 20:02:58

* [python](http://so.csdn.net/so/search/s.do?q=python&t=blog) /
* [智慧優化演算法](http://so.csdn.net/so/search/s.do?q=%E6%99%BA%E8%83%BD%E4%BC%98%E5%8C%96%E7%AE%97%E6%B3%95&t=blog) /
* [deap](http://so.csdn.net/so/search/s.do?q=deap&t=blog) /
* [遺傳演算法](http://so.csdn.net/so/search/s.do?q=%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95&t=blog) /
* [背包問題](http://so.csdn.net/so/search/s.do?q=%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98&t=blog)

**特殊**

**自訂評價函數**

同前

def evalKnapsack(individual):

weight = 0.0

value = 0.0

for item in individual:

weight += items[item][0]

value += items[item][1]

if len(individual) > MAX\_ITEM or weight > MAX\_WEIGHT:

return 10000, 0 # Ensure overweighted bags are dominated

return weight, value,

**自訂交叉函數**

def cxSet(ind1, ind2):

"""Apply a crossover operation on input sets. The first child is the

intersection of the two sets, the second child is the difference of the

two sets.

"""

temp = set(ind1) # Used in order to keep type

ind1 &= ind2 # Intersection (inplace)

ind2 ^= temp # Symmetric Difference (inplace)

return ind1, ind2

&=,^= python中的位運算子   
*建議在新標籤頁打開圖片*   


**自訂變異函**

def mutSet(individual):

"""Mutation that pops or add an element."""

if random.random() < 0.5:

if len(individual) > 0: # We cannot pop from an empty set

individual.remove(random.choice(sorted(tuple(individual))))

else:

individual.add(random.randrange(NBR\_ITEMS))

return individual,

**使用短版本的遺傳演算法**

def main():

random.seed(64)

NGEN = 50

MU = 50

LAMBDA = 100

CXPB = 0.7

MUTPB = 0.2

pop = toolbox.population(n=MU)

hof = tools.ParetoFront()

stats = tools.Statistics(lambda ind: ind.fitness.values)

stats.register("avg", numpy.mean, axis=0)

stats.register("std", numpy.std, axis=0)

stats.register("min", numpy.min, axis=0)

stats.register("max", numpy.max, axis=0)

algorithms.eaMuPlusLambda(pop, toolbox, MU, LAMBDA, CXPB, MUTPB, NGEN, stats,

halloffame=hof)

return pop, stats, hof

此處與之前的文章效果類似   
- [粒子群優化演算法](http://blog.csdn.net/fontthrone/article/details/78253284)   
- [短版本可以參考官網介紹](http://deap.gel.ulaval.ca/doc/default/examples/ga_onemax_short.html)

**原始程式碼**

#!usr/bin/env python

#-\*- coding:utf-8 \_\*-

"""

@author:fonttian

@file: knapsackProblem.py

@time: 2017/10/15

"""

# This file is part of DEAP.

#

# DEAP is free software: you can redistribute it and/or modify

# it under the terms of the GNU Lesser General Public License as

# published by the Free Software Foundation, either version 3 of

# the License, or (at your option) any later version.

#

# DEAP is distributed in the hope that it will be useful,

# but WITHOUT ANY WARRANTY; without even the implied warranty of

# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

# GNU Lesser General Public License for more details.

#

# You should have received a copy of the GNU Lesser General Public

# License along with DEAP. If not, see <http://www.gnu.org/licenses/>.

import random

import numpy

from deap import algorithms

from deap import base

from deap import creator

from deap import tools

IND\_INIT\_SIZE = 5 # 基因編碼位元數

MAX\_ITEM = 50

MAX\_WEIGHT = 50

NBR\_ITEMS = 20

# To assure reproductibility, the RNG seed is set prior to the items

# dict initialization. It is also seeded in main().

random.seed(64)

# Create the item dictionary: item name is an integer, and value is

# a (weight, value) 2-uple.

items = {}

# Create random items and store them in the items' dictionary.

for i in range(NBR\_ITEMS):

items[i] = (random.randint(1, 10), random.uniform(0, 100))

creator.create("Fitness", base.Fitness, weights=(-1.0, 1.0))

creator.create("Individual", set, fitness=creator.Fitness)

toolbox = base.Toolbox()

# Attribute generator

toolbox.register("attr\_item", random.randrange, NBR\_ITEMS)

# Structure initializers

toolbox.register("individual", tools.initRepeat, creator.Individual,

toolbox.attr\_item, IND\_INIT\_SIZE)

toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evalKnapsack(individual):

weight = 0.0

value = 0.0

for item in individual:

weight += items[item][0]

value += items[item][1]

if len(individual) > MAX\_ITEM or weight > MAX\_WEIGHT:

return 10000, 0 # Ensure overweighted bags are dominated

return weight, value,

def cxSet(ind1, ind2):

"""Apply a crossover operation on input sets. The first child is the

intersection of the two sets, the second child is the difference of the

two sets.

"""

temp = set(ind1) # Used in order to keep type

ind1 &= ind2 # Intersection (inplace)

ind2 ^= temp # Symmetric Difference (inplace)

return ind1, ind2

def mutSet(individual):

"""Mutation that pops or add an element."""

if random.random() < 0.5:

if len(individual) > 0: # We cannot pop from an empty set

individual.remove(random.choice(sorted(tuple(individual))))

else:

individual.add(random.randrange(NBR\_ITEMS))

return individual,

toolbox.register("evaluate", evalKnapsack)

toolbox.register("mate", cxSet)

toolbox.register("mutate", mutSet)

toolbox.register("select", tools.selNSGA2)

def main():

random.seed(64)

NGEN = 50

MU = 50

LAMBDA = 100

CXPB = 0.7

MUTPB = 0.2

pop = toolbox.population(n=MU)

hof = tools.ParetoFront()

stats = tools.Statistics(lambda ind: ind.fitness.values)

stats.register("avg", numpy.mean, axis=0)

stats.register("std", numpy.std, axis=0)

stats.register("min", numpy.min, axis=0)

stats.register("max", numpy.max, axis=0)

algorithms.eaMuPlusLambda(pop, toolbox, MU, LAMBDA, CXPB, MUTPB, NGEN, stats,

halloffame=hof)

return pop, stats, hof

if \_\_name\_\_ == "\_\_main\_\_":

pop, stats, hof = main()

print("最佳裝包為(最佳個體) :",hof[-1])

print(len(pop))

print(len(hof))

print("最佳裝包時的重量與價值(最佳適應度) :",evalKnapsack(hof[-1]))

版權聲明：如果您覺得本文不錯,麻煩滑鼠右轉點下贊,謝謝.如果有什麼問題,請直接留言.轉載只需標明原文網址即可. https://blog.csdn.net/FontThrone/article/details/78253539

* 本文已收錄於以下專欄：
* [劍指數據科學](https://blog.csdn.net/column/details/16266.html)