

Oct 30, 14 8:05

csc710sbse: hw4:Rahul Krishna

Page 1/1

```
"""
rDiv Demo
"""

5  import sk
   import random

   rand=random.uniform
   randi=random.randint

10  rdivDemo=sk.rdivDemo

   Range1=[rand(0,4) for _ in xrange(5)]; Range1.insert(0,'Range1')
   Range2=[rand(3,13) for _ in xrange(5)]; Range2.insert(0,'Range2')
15  Range3=[rand(7,17) for _ in xrange(5)]; Range3.insert(0,'Range3')
   Range4=[rand(10,27) for _ in xrange(5)]; Range4.insert(0,'Range4')
   Range5=[rand(12,30) for _ in xrange(5)]; Range5.insert(0,'Range5')
   Range6=[rand(30,50) for _ in xrange(5)]; Range6.insert(0,'Range6')

20  def _rdiv():
      rdivDemo([
          Range1,
          Range2,
25      Range3,
          Range4,
          Range5,
          Range6,
      ])

30  _rdiv()
```

Oct 30, 14 7:49

csc710sbse: hw4:Rahul Krishna

Page 1/2

```

# -*- coding: utf-8 -*-
"""
Created on Mon Sep 15 03:04:43 2014

5  @author: rkrns
"""
from __future__ import division
import sys
import math, random, numpy as np, scipy as sp
10 sys.dont_write_bytecode = False
from models import *
from anzeigen import *
from dynamikliste import *
# from sk import Num
15 import analyzer

# Define some aliases.
rand = random.uniform
randi = random.randint
20 exp = math.exp

class SimulatedAnnealer(object):
    "SA"
    def __init__(self, modelName, emax, emin, disp=False, early=False):
25         self.modelName = modelName
        self.disp = disp
        self.early = early
        self.emax, self.emin = emax, emin
    def runSearcher(self):
30         modelbasics = modelBasics(self.modelName);
        modelFunction = self.modelName()
        anz = anzeigen();
        hi, lo, kooling, indepSize, iterations = \
            modelFunction.eigenschaften()
35         emax, emin = self.emax, self.emin
        sb = s = [randi(lo, hi) for z in xrange(indepSize)];
        eb = e = modelbasics.energy(s, self.emax, self.emin)
        enRec = dynamikliste() # Creates a growing list.
        enRec[0] = 0;
40         # Since iterations start from 1, lets initialize enRec[0] to 0
        analyser = analyzer.analyser()
        epochs = 3 if self.early else iterations;
        k = 1;
        while epochs ^ k < iterations:
45             sn = modelbasics.neighbour(s, hi, lo)
            en = modelbasics.energy(sn, emax, emin)
            t = k / iterations
            if en < eb:
                eb, sb, enRec[k] = en, sn, en;
50             #if self.disp:
                #modelbasics.say('!!')
            if en < e:
                s, e, enRec[k] = sn, en, en;
                #if self.disp:
                    #modelbasics.say('++')
55
            if modelbasics.do_a_randJump(en, e, t, kooling):
                # The cooling factor needs to be really low for some reason!!
                s, e, enRec[k] = sn, en, en;
                #if self.disp:
                    #modelbasics.say('??')
60
            else:
                enRec[k] = en
                #if self.disp:
                    # modelbasics.say('.')
65             if k % 50 == 0 ^ k > 50:
                # print enRec[:-10]
                proceed = analyser.isItGettinBetter(enRec[k - 100:])
                if proceed:
                    epochs += 1;
                else:
                    epochs -= 1;
                    # print enRec[k-40:] #
                    k = k + 1
75             era=1;

# Print Energy and best value.
for i in xrange(k):
    if self.disp:
80         if i % 50 == 0:
            print era, anz.xtile(enRec[i - 50:], show='%0.2E')
            era+=1
        if self.disp:

```

Oct 30, 14 7:49

csc710sbse: hw4:Rahul Krishna

Page

```

        modelbasics.say('\n')
85         return [eb, modelbasics.energyIndv(sb, emax, emin)]

class MaxWalkSat(object):
    "MWS"
    def __init__(self, modelName, emax, emin, disp=False, early=True, maxTries=100,
90                 maxChanges=100):
        self.modelName = modelName
        self.disp = disp
        self.maxTries = maxTries
        self.maxChanges = maxChanges
95         self.emax, self.emin = emax, emin
    def runSearcher(self):
        modelbasics = modelBasics(self.modelName);
        modelFunction = self.modelName()
        hi, lo, kooling, indepSize, iterations = \
100         modelFunction.eigenschaften()
        thresh=1e-4
        emax, emin = self.emax, self.emin
        for i in xrange(self.maxTries):
            # Lets create a random assignment, I'll use list comprehesions here.
105             x = xn = xb = [rand(lo, hi) for z in xrange(indepSize)]
            # Create a threshold for energy,
            # let's say thresh=0.1% of emax (which is 1) for starters
            for j in xrange(self.maxChanges):
                # Let's check if energy has gone below the threshold.
110                 # If so, look no further.
                if modelbasics.energy(xn, emax, emin) < thresh:
                    xb=xn
            else:
                # Choose a random part of solution x
                randIndx = randi(0, indepSize - 1)
115                 if rand(0, 1) > 1 / (indepSize + 1): # Probablity p=0.33
                    y = xn[randIndx]
                    xn[randIndx] = modelbasics.simpleneighbour(y, hi, lo)
                    # print 'Random change on', randIndx
120                 else:
                    # xTmp is a temporary variable
                    xBest = emax;
                    # Step from xmin to xmax, take 10 steps
                    Step = np.linspace(lo, hi, 10)
125                     for i in xrange(np.size(Step)):
                        xNew = xn; xNew[randIndx] = Step[i];
                        if modelbasics.energy(xNew, emax, emin) < xBest:
                            xBest = modelbasics.energy(xNew, emax, emin)
                            xn = xNew
130
                if modelbasics.energy(xn, emax, emin) < modelbasics.energy(xb,
                                                                    emax,
                                                                    emin):
                    xb = xn
135                     print modelbasics.energy(xn, emax, emin)
        return [modelbasics.energy(xb, emax, emin), modelbasics.energyIndv(xb, emax, emin)]

if __name__ == '__main__':
140     SimulatedAnnealer(Schaffer)

```

Oct 30, 14 8:12

csc710sbse: hw4:Rahul Krishna

Page 1/1

```

from __future__ import division
from searcher import *
from models import *
import sys
5 from decimal import *
import numpy as np
from anzeigen import *
from time import gmtime, strftime
import sk
10 import sys, random, math, datetime, time, re
from base import xtile
sys.dont_write_bytecode = True
rseed=random.seed
rdivdemo=sk.rdivDemo
15 #=====
# Baselineing
#=====
emin=10**32;
emax=-10**32;
20 baselining = {}

for x in [Schaffer, Kursawe,
          Fonseca, ZDT1, ZDT3, Viennet3]:
    baselining.update({x.__doc__: (0, 0)})
25 rseed(1)
    for y in [SimulatedAnnealer, MaxWalkSat]:
        k=modelBasics(x)
        eMax, eMin = k.baselining(x)
        (emax, emin) = baselining[x.__doc__]
30 emax= eMax if eMax>emax else emax
        emin= eMin if eMin<emin else emin
        baselining.update({x.__doc__: (emax, emin)})

35 for x in [ZDT1]:
    rseed(1)
    early=True
    eb=30*[0]
    eb1=30*[0]
40 eb2=30*[0]
    (e1, e2)= baselining[x.__doc__]
    for y in [SimulatedAnnealer, MaxWalkSat]:
        print 'Model: ', x.__doc__
        print 'Searcher: ', y.__doc__
45 print strftime("%a, %d %b %Y %H:%M:%S ", gmtime()), '\n'
        k=x()
        reps=30
        dsp1=anzeigen();
        E=[]
50 E1=[]
        E2=[]
        hi, lo, kooling, indepSize, iterations = k.eigenschaften()
        print 'Settings:'
        print 'min=', lo, ', max=', hi, ', Cooling Factor=', kooling, '\n'
55 if early: print 'Early Termination!', '\n'
        for r in xrange(reps):
            a=y(x, e1, e2, disp=False, early=early)
            eTmp = a.runSearcher()
            E.append([eTmp[0]])
60 #E1.append(eb1[r])
            #E2.append(eb2[r])
            #print dsp1.xtile(eb[1:])
            "" for r in xrange(4):
            print dsp1.xtile(eb[rr+50], lo=lo, hi=hi)""
65 E.insert(0, y.__doc__)
            print E
            #E1.insert(0, 'Objective 1')
            #E2.insert(0, 'Objective 2')
            print 'Best Energy: ', "{:3F}".format(Decimal(str(np.sum(eb)/reps))), '\n'
70
def _rDiv():
    rdivdemo(E)
_rDiv()
for _ in xrange(50): sys.stdout.write('_')
75 print '\n'

#

80 sys.stdout.write('\n')

```

Oct 30, 14 2:37

csc710sbse: hw4:Rahul Krishna

Page 1/3

```

"""
A models file that can be imported to run optimizers
"""
from __future__ import division
5 import sys
import math, random, numpy as np, scipy as sp
sys.dont_write_bytecode = False
# Define some aliases.
rand=random.uniform
10 randi=random.randint
exp=math.e
sin=math.sin
sqrt=math.sqrt
pi=math.pi

15 class modelBasics(object):
    def __init__(i,model):
        i.model=model()
        i.name=model.__name__
    def do_a_randJump(i, e, en, t, k):
        p=exp*(-(e-en)/(t**k))>rand(0,1)
        return p
    def simpleneighbour(self,x,xmax,xmin):
        return xmin+(xmax-xmin)*rand(0,1)
    def neighbour(i,x,xmax,xmin):
25     def __new(x,z):
        return xmin+(xmax-xmin)*rand(0,1) if rand(0,1)<1/(i.model.indepSize) else x[z]
        x_new=[__new(x,z) for z in xrange(i.model.indepSize)]
        return x_new
    def energy(i,x,emax,emin):
        ener=i.model.score(x)
        e_norm=(ener-emin)/(emax-emin)
        return e_norm
    def energyIndv(i,x,emax,emin):
        ener=i.model.eachObjective(x)
        e_norm=[abs((e-(emin))/(emax-emin)) for e in ener]
        return e_norm
    def baselining(i,model):
        emax=-10**32;emin=10**32;
        indepSize=i.model.indepSize;
        for _ in xrange(1000):
            x_tmp=[rand(i.model.baselo,i.model.basehi) for _ in xrange(indepSize)]
            ener=i.model.score(x_tmp);
            if ener>emax:
                emax=ener
            elif ener<emin:
                emin=ener
        return emax,emin
50 f=open('log_sa_schaffer.txt','w')
    def say(i,x):
        sys.stdout.write(str(x));
        sys.stdout.flush()

class Schaffer(object):
55     "Schaffer"
    def __init__(i,hi=100,lo=-100, basehi=1000, baselo=-1000, kooling=1e-4, indepSize=1, iterations=2000
):
        i.hi, i.lo, i.basehi, i.baselo, i.kooling, i.indepSize, i.iterations= hi, lo, basehi, baselo, kool
ing, indepSize, iterations

    def f1(i,x):
        return x*x
    def f2(i,x):
        return (x-2)**2
    def score(i,x):
        return i.f1(x[0])+i.f2(x[0])
65     def eachObjective(i,x):
        return [i.f1(x[0]), i.f2(x[0])]
    def eigenschaften(i):
        return i.hi, i.lo, i.kooling, i.indepSize, i.iterations

70 class Kursawe(object):
    "Kursawe"
    def __init__(i,hi=5,lo=-5,kooling=0.6, a=0.8, b=3, indepSize=3, basehi=5,
        baselo=-5, thresh=1e-2, iterations=2000):
        i.hi, i.lo, i.basehi, i.baselo, i.kooling = hi, lo, basehi, baselo, kooling
75 i.thresh=thresh
        i.a, i.b, i.indepSize, i.iterations= a, b, indepSize, iterations

    def f1(i,x):
        return np.sum([-10*exp**(-0.2*sqrt(x[z]**2+x[z+1]**2)) \
80         for z in xrange(i.indepSize-1)])
    def f2(i,x):

```

Oct 30, 14 2:37

csc710sbse: hw4:Rahul Krishna

Page

```

        return np.sum([abs(x[z])**i.a+5*sin(x[z]**i.b) \
        for z in xrange(i.indepSize)])
    def score(i,x):
        return i.f1(x)+i.f2(x)
85     def eachObjective(i,x):
        return [i.f1(x), i.f2(x)]
    def eigenschaften(i):
        return i.hi, i.lo, i.kooling, i.indepSize, i.iterations

90 class Fonseca(object):
    "Fonseca"
    def __init__(i,hi=4,lo=-4, basehi=4, baselo=-4, kooling=1.99, indepSize=3,
        thresh=1e-2, iterations=2000):
95         i.hi, i.lo, i.basehi, i.baselo, i.kooling, i.indepSize, i.thresh, i.iterations= \
            hi, lo, basehi, baselo, kooling, indepSize, thresh, iterations

    def f1(i,x):
        return (1-exp**np.sum([(x[z]-1)/((i.indepSize)**0.5)]) \
100         for z in xrange(i.indepSize)))
    def f2(i,x):
        return (1-exp**np.sum([(x[z]+1)/((i.indepSize)**0.5)]) \
        for z in xrange(i.indepSize)))
    def score(i,x):
        return i.f1(x)+i.f2(x)
    def eachObjective(i,x):
        return [i.f1(x), i.f2(x)]
    def eigenschaften(i):
        return i.hi, i.lo, i.kooling, i.indepSize, i.iterations

110 class ZDT1(object):
    "ZDT1"
    def __init__(i,hi=1,lo=0, basehi=1, baselo=0, kooling=7e-3, indepSize=30, iterations=2000):
        i.hi, i.lo, i.basehi, i.baselo, i.kooling, i.indepSize, i.iterations= hi, lo, basehi, baselo, kool
ing, indepSize, iterations

115     def f1(i,x):
        return x[0]
    def g(i,x):
        return (1+9*(np.sum(x[1:]))/(i.indepSize-1))
    def f2(i,x):
        return i.g(x)*(1-sqrt(x[0]/i.g(x)))
    def score(i,x):
        return i.f1(x)+i.f2(x)
    def eachObjective(i,x):
        return [i.f1(x), i.f2(x)]
125     def eigenschaften(i): # German for features
        return i.hi, i.lo, i.kooling, i.indepSize, i.iterations

class ZDT3(object):
130     "ZDT3"
    def __init__(i,hi=1,lo=0, basehi=2, baselo=0, kooling=7e-3, indepSize=30,
        thresh=1e-2, iterations=2000):
        i.hi, i.lo, i.basehi, i.baselo, i.thresh = hi, lo, basehi, baselo, thresh
        i.kooling, i.indepSize, i.iterations = kooling, indepSize, iterations
135     def f1(i,x):
        return x[0]
    def g(i,x):
        return (1+9*(np.sum(x[1:]))/(i.indepSize-1))
    def f2(i,x):
        return i.g(x)*(1-(x[0]/i.g(x))**0.5-(x[0]/i.g(x))*sin(10*math.pi*x[0]))
140     def score(i,x):
        return (i.f1(x)+i.f2(x))
    def eachObjective(i,x):
        return [i.f1(x), i.f2(x)]
145     def eigenschaften(i): # German for features
        return i.hi, i.lo, i.kooling, i.indepSize, i.thresh, i.iterations

class Viennet3(object):
    "Viennet3"
150     def __init__(i,hi=1,lo=0, basehi=1, baselo=0, kooling=7e-3, indepSize=2, iterations=2000):
        i.hi, i.lo, i.basehi, i.baselo, i.kooling, i.indepSize, i.iterations= hi, lo, basehi, baselo, kool
ing, indepSize, iterations

    def f1(i,x):
        return 0.5*x[0]**2+x[1]**2+sin(x[0]**2+x[1]**2)
155     def f2(i,x):
        return (3*x[0]-2*x[1]+4)**2/8+(x[0]-x[1]+1)**2/175+15
    def f3(i,x):
        return 1/(x[0]+x[1]+1)-1.1*exp**(-x[0]**2-x[1]**2)
    def score(i,x):
        return i.f1(x)+i.f2(x)+i.f3(x)
160     def eachObjective(i,x):
        return [i.f1(x), i.f2(x)]

```

Oct 30, 14 2:37

csc710sbse: hw4:Rahul Krishna

Page 3/3

```
def eigenschaften(i): # German for features  
    return i.hi, i.lo, i.kooling, i.indepSize, i.iterations
```