

Oct 14, 14 10:17

csc791sbse:hw7:Fu

Page 1/1

```

from __future__ import division
import sys, random, math
from sk import *
from sa import *
5 from mws import *
from ga import *
from de import *
from pso import *
sys.dont_write_bytecode = True

10
@demo
def HW7():
    for klass in [ Schaffer, Fonseca, Kursawe, ZDT1, ZDT3, Viennet3, DTLZ7, Schwefe
1]]:
        # for klass in [ Schaffer]:
15         print "\n!!!!", klass.__name__
        alleB = []
        #searcher = {"ga":ga}
        searcher = {"sa":sa, "mws":mws, "ga":ga, "de": de, "pso":pso}
        Settings.other.repeats = 1
        for key in searcher.keys():
20             repeats = 5
             eb = repeats*[0]
             name = klass.__name__
             reseed()
25             for r in range(repeats):
                 results=searcher[key](klass()) # lohi is a list containing [lo,hi] paris
of fl&f2
                 eb[r] = results[0] if isinstance(results, tuple) else results
                 eb.insert(0, key)
                 allEB.append(eb)
30             rdivDemo(allEB)

@demo
def testmodel():
    # model = ZDT3()
35    model = Schaffer()
    depen = model.getDepen(model.generate_x())
    print depen

if __name__ == "__main__": eval(cmd())
40
45

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 1/2

```

from __future__ import division
from log import *
from models import *
from xtile import *
5 from base import *
import sys, random, math, datetime, time, re, pdb, operator
sys.dont_write_bytecode = True

@printlook
10 def ga(model):
    mutationRate = 1/model.n
    population = []
    solution = []
    children = []
15    fitness = {}
    history = {}
    mateNum = 20
    def selection(sortedFitness):
        return [population[sortedFitness[0][0]], population[sortedFitness[1][0]]] #
        # sorted[0] and [1] are the smallest two we preferred
20    def crossover(selected):
        '''crossover will do this way: offspring1 = p* parent 1+ (1-p)* parent2 for numbers between two points '''
        def what(lst):
            return lst[0] if isinstance(lst, list) else lst
        children1 = []
25        if rand() > Settings.ga.crossRate:
            return selected[0]
        else:
            if model.n == 1:
                children1 = [(what(selected[0]) + what(selected[1]))*0.5]
30            else:
                index = sorted([random.randint(0, model.n - 1) for _ in xrange(Settings.
                ga.crossPoints)])
                parent1 = selected[0]
                parent2 = selected[1]
                children1 = parent1[:]
35                children1[index[0]:index[1]] = parent2[index[0]:index[1]]
            return children1
        def mutate(children, selected):
            # print children
            for k, n in enumerate(children):
40                if rand() < mutationRate:
                    children[k] = selected[random.randint(0,1)][random.randint(0, model.n-1)]
            # pick value from mom or dad
            # print children
            return children
        def tournament(sortedFitness, m=10): # do tournament selection, select the best
            # daddy or mom in m = 10 candidates
45            index = []
            for _ in range(m):
                index.append(random.randint(0, Settings.ga.pop-1))
            betterIndex = list(set(sorted(index)))
            parentlst = [population[sortedFitness[betterIndex[0]][0]], population[sorted
            Fitness[betterIndex[1]][0]]]
50            return parentlst
        def fit(fitness):
            sortedFitness = sorted(fitness.items(), key = lambda x:x[1]) # a sorted list
            return sortedFitness[:Settings.ga.pop] # just return the top 50 candidates a
            s new populatioin
        def produce(selected):
55            children = crossover(selected)
            children = mutate(children, selected)
            return children

min_energy, max_energy = model.baseline()
60 eb= 0
solution = []
control = Control(model, history)
for _ in xrange(Settings.ga.pop):
    temp = model.generate_x()
    population.append(temp)
65 # for num in Settings.ga.genNum:

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 2/2

```

t = 0
while(t < Settings.ga.genNum): # figure stop out
    stopsign = control.next(t) #true ---stop
70    if stopsign:
        break
    for (k, xlst) in enumerate(population):
        fitness[k] = model.getDepon(xlst)
        newpopfitness = fit(fitness)
75    for n, k in newpopfitness:
        population[n] = population[newpopfitness[0][0]] # new generation
        control.logxy(population[n])
        # for n, k in population:
        #     control.logxy(k) # log new generation
80    eb = model.norm(newpopfitness[0][1])
    solution = population[newpopfitness[0][0]]
    for _ in range(mateNum):
        selected = tournament(newpopfitness)
        children.append(produce(selected))
85    population.extend(children)
    t +=1
    print "best solution: %s" % str(solution)
    print "best normalized results: %s" % str(eb)
    print "-"*20
90    # printReport(model)
    # lohi=printRange(model)
    # return eb,lohi
    if Settings.other.xtile:
        printReport(model, history)
95    print "\n"
    printSumReport(model, history)
    if Settings.other.reportrange:
        rrange=printRange(model, history)
        return eb,rrange
100    else:
        return eb

def startga():
    for klass in [Schaffer, Fonseca, Kursawe, ZDT1, ZDT3, Viennet3]:
105    # for klass in [DTLZ7]:
        print "-"*50
        print "!!!!", klass.__name__,
        print "\nSearcher: GA"
        reseed()
110    ga(klass())

if __name__ == "__main__":startga()
    # print sortedFitness
115
120

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 1/1

```

from __future__ import division
import sys, random, math
from models import *
from base import *
5 #this is a test
sys.dont_write_bytecode = True
@printlook
def sa(model):
    def P(old, new, t):
10         prob = math.e**((old - new)/(t+0.00001))
        return prob
    history = {}
    eb = 0.0
    for _ in xrange(Settings.other.repeats):
15         #reseed()
        min_energy, max_energy = model.baseline()
        s = model.generate_x()
        e = model.norm(model.getDepen(s))
        sb = s[:]
20         eb = e
        k = 1
        icontrol = Control(model, history)
        while k < Settings.sa.kmax:
            stopsign = icontrol.next(k) #true ---stop
25             if stopsign:
                 break
            sn = model.sa_neighbor(s)
            en = model.norm(model.getDepen(sn))
            icontrol.logxy(sn)
30             temp = (k/Settings.sa.kmax)**Settings.sa.cooling
            if en < eb:
                sb = sn[:] ###!!!! can't do sb = sn for lists, because
                eb = en
                if Settings.other.show: say('!!')
35             if en < e:
                s = sn[:]
                e = en
                if Settings.other.show:say('+')
            elif P(e, en, temp) < random.random():
40                 s = sn[:]
                e = en
                if Settings.other.show:say('?')
                if Settings.other.show:say('.')
                k = k + 1
45             if k % 30 == 0:
                if Settings.other.show:print "\n"
                if Settings.other.show:say(str(round(eb,3)))
        if Settings.other.xtile:
            printReport(model, history)
50             print "\n"
            printSumReport(model, history)
            # print "\n-----\n:Normalized Sum of Objectives : ",str(round(eb,3)),"\n:Solu
            tion",sb
            if Settings.other.reportrange:
                rrange=printRange(model, history)
55             return eb,rrange
    else:
        return eb

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 1/1

```

from __future__ import division
import sys, random, math
from models import *
from base import *
5 sys.dont_write_bytecode = True
@printlook
def mws(model):
    norm_energy = 0
    eraScore = []
10    control = Control(model)
    optimalsign = False
    eb = 0.0
    norm_energy = 10
    history = {}
15    for _ in xrange(Settings.other.repeats):
        min_energy, max_energy = model.baseline()
        control = Control(model, history)
        total_changes = 0
        total_tries = 0
20    for k in xrange(Settings.mws.max_tries):
        if control.lives == 0:
            break
        solution = model.generate_x()
        total_tries += 1
25    for _ in range(Settings.mws.max_changes):
        stopsign = control.next(total_changes) #true ---stop
        if stopsign:
            break
        norm_energy = model.norm(model.getDepen(solution))
30    if norm_energy < Settings.mws.threshold:
        optimalsign = True
        break
        if random.random() < Settings.mws.prob:
            solution[random.randint(0,model.n-1)] = model.generate_x()[random.rand
int(0,model.n-1)]
35    control.logxy(solution)
        if Settings.other.show: say("+")
        else:
            solution = model.mws_neighbor(solution)
            control.logxy(solution)
40    if Settings.other.show: say("!")
        if Settings.other.show: say(".")
        if total_changes % 30 == 0:
            if Settings.other.show: print "\n"
            if Settings.other.show: say(str(round(model.norm(model.getDepen(solutio
n)), 3)))
45    total_changes += 1
        # if optimalsign or k == Settings.mws.max_tries-1:
        if Settings.other.xtile:
            say("\n")
            say(str(round(model.norm(model.getDepen(solution)), 3)))
50    print "\n"
        printReport(model, history)
        print "\n"
        printSumReport(model, history)
        if Settings.other.reportrange:
            rrange = printRange(model, history)
55    return norm_energy, rrange
    else:
        return norm_energy

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 1/2

```

from __future__ import division
import sys, random, math
from models import *
from base import *

5 def de(model):
    eb = 0.0
    np = Settings.de.np
    repeats = Settings.de.repeats
10 fa = Settings.de.f
    cr = Settings.de.cr
    threshold = Settings.de.threshold
    min_e, max_e = model.baseline()
    # s = model.generate_x()
15 # e = model.norm(model.getDepen(s))
    # sb = s[:]
    # eb = e
    indices = []
    scores = {}
20 def evaluate(pop):
    for n, x in enumerate(pop):
        scores[n] = model.norm(model.getDepen(x))
        # print scores
    ordered = sorted(scores.items(), key=lambda x: x[1]) # alist of tuple
25 # print ordered
    return pop[ordered[0][0]], ordered[0][1]
def gen3(n,f,frontier):
    seen = [n]
    def genl(seen):
30         while 1:
            k = random.randint(0, np -1)
            if k not in seen:
                seen += [k]
                break
35         return frontier[k]
    a = genl(seen)
    b = genl(seen)
    c = genl(seen)
    return a, b, c
40 def trim(x):
    return max(model.lo, min(x,model.hi))

def update(n,f,frontier):
    newf = []
45 a, b, c = gen3(n,f,frontier)
    for n in xrange(len(f)):
        if cr < rand():
            newf.append(f[n])
        else:
50             newf.append(trim(a[n]+fa*(b[n]-c[n])))
    return newf

frontier = [model.generate_x() for _ in xrange(np)]
sb, eb = evaluate(frontier)
55 for k in xrange(repeats):
    if eb < threshold:
        break
    nextgen = []
    for n,f in enumerate (frontier):
60         new = update(n, f, frontier)
        if model.norm(model.getDepen(new)) < model.norm(model.getDepen(f)):
            nextgen.append(new)
        else:
            nextgen.append(f)
65 frontier = nextgen
    sb, eb = evaluate(frontier)
print eb
if Settings.other.reportrange:
    xrange=printRange(model, history) # no history right now!
70 return eb, xrange
else:
    return eb

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 2/2

```

def deDemo():
75     for klass in [Schaffer]:
        # for klass in [DTLZ7]:
            print "="*50
            print "!!!!", klass.__name__,
            print "\nSearcher: DE"
80     reseed()
    de(klass())
if __name__ == "__main__": deDemo()

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 1/5

```

from __future__ import division
from log import *
import sys, random, math, datetime, time, re, pdb
sys.dont_write_bytecode = True

exp = math.e
sqrt = math.sqrt
sin = math.sin
10 cos = math.cos
pi = math.pi

class Model:
    def name(i):
        return i.__class__.__name__
    def setup(i):
        i.min = 10**(5)
        i.max = -10**(5)
        i.xy = Options(x = [i.generate_x()], y = [i.f1, i.f2])
        i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range
20 (i.fn)]) # hardcoded 2
        i.history = {} # hold all logs for eras
    def generate_x(i):
        x = [i.lo + (i.hi-i.lo)*random.random() for _ in range(i.n)]
        return x
    def getDepen(i, xlst):
        # y = [i.f1, i.f2]
        return sum([f(xlst) for f in i.xy.y])
    def getDepenlst(i, xlst):
        return [f(xlst) for f in i.xy.y]
    def cloneModel(i): # from Dr.Menzies'
        return i.__class__()
    def logxy(i, x):
        for val, log in zip(x, i.log.x): log += val
        y = i.getDepenlst(x)
        for val, log in zip(y, i.log.y): log += val
    def better(news, olds): # from Dr.Menzies'
    def worsed():
        return ((same ^ ¬ betterIqr) ∨
40             (¬ same ^ ¬ betterMed))
    def bettered():
        return ¬ same ^ betterMed
        out = False
        for new, old in zip(news.log.y, olds.log.y):
            betterMed, same, betterIqr = new.better(old)
            # print betterMed, same, betterIqr
            # pdb.set_trace()
            if worsed(): : return False # never any worsed
            if bettered(): out= out ∨ True # at least one bettered
        return out
    def sa_neighbor(i, old):
        p = 1/i.n
        new = old[:]
        for j in range(len(old)):
            if random.random() < p:
55         new_gen = i.generate_x()
            new[j] = new_gen[random.randint(0, i.n-1)]
        return new
    def mws_neighbor(i, solution):
        optimized_index = random.randint(0, len(solution)-1)
        increment = (i.hi - i.lo)/10
        temp_min = i.norm(i.getDepen(solution))
        temp_solution = solution[:]
        # print "old solution : %s" % solution
        # print "old norm energy : %s" % i.norm(i.getDepen(solution))
        for _ in range(10):
65         temp_solution[optimized_index] = i.lo + increment
            temp = i.norm(i.getDepen(temp_solution))
            if temp < temp_min:
                temp_min = temp
            solution = temp_solution[:]
70         # print "new solution : %s" % solution
            # print "new norm energy : %s" % i.norm(i.getDepen(solution))

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 2/5

```

        return solution
    def baseline(i):
        # model = eval(model+"()")
75     for _ in xrange(Settings.other.baseline):
        temp = i.getDepen(i.generate_x())
        if temp > i.max:
            i.max = temp
        if temp < i.min:
80         i.min = temp
        return i.min, i.max
    def norm(i, x):
        e = (x - i.min)/(i.max - i.min)
85     return max(0, min(e, 1)) #avoid values <0 or >1

class Control(object): # based on Dr.Menzies' codes
    def __init__(i, model, history = None):
        i.kmax = Settings.sa.kmax
        i.era = Settings.other.era
90     i.lives = Settings.other.lives
        i.history = {} if history == None else history
        i.logAll = {}
        i.model = model
    def __call__(i, k):
        i.next(k)
    def logxy(i, results):
        both = [i.history, i.logAll, i.model.history]
        for log in both:
100         if ¬ i.era in log:
            log[i.era] = i.model.cloneModel()
        for log in both:
            log[i.era].logxy(results)
    def checkimprove(i):
105     if len(i.logAll) ≥ 2:
        current = i.era
        before = i.era - Settings.other.era
        currentLog = i.logAll[current]
        beforeLog = i.logAll[before]
        # pdb.set_trace()
        if ¬ currentLog.better(beforeLog):
110         pass
        else:
            i.lives += 1
    def next(i, k):
        if k ≥ i.era:
            i.checkimprove()
            i.era += Settings.other.era
            if i.lives == 0:
120             return True
        else:
            i.lives -= 1
            return False

'''Schaffer'''
class Schaffer(Model):
    def __init__(i):
130     i.lo = -5
        i.hi = 5
        i.n = 1
        i.fn = 2
        i.setup()
    def f1(i, x):
        return x[0] * x[0]
    def f2(i, x):
        return (x[0]-2) ** 2

'''Fonseca'''
class Fonseca(Model):
    def __init__(i):
140     i.lo = -4
        i.hi = 4
        i.n = 3
145

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 3/5

```

    i.fn = 2
    i.setup()
    # def f1(i, xlst):
    #     return (1 - exp**(-1 * sum([(xlst[k] - 1/sqrt(i.n))**2 for k in xrange(i.n
    ])))
150 # def f2(i, xlst):
    #     return (1 - exp**(-1 * sum([(xlst[k] + 1/sqrt(i.n))**2 for k in xrange(i.n
    ])))
    def f1(i, xlst):
        def f1_sum(x_list, n):
            value = []
155         for item in x_list:
            value.append((item - 1/math.sqrt(n))**2)
            return sum(value)
        return 1 - math.e ** (-1* f1_sum(xlst, i.n))
    def f2(i,xlst):
160     def f2_sum(x_list, n):
        value = []
        for item in x_list:
            value.append((item + 1/math.sqrt(n))**2)
            return sum(value)
165     return 1 - math.e ** (-1* f2_sum(xlst, i.n))

'''Kusarvs'''
class Kursawe(Model):
    def __init__(i):
170     i.lo = -5
        i.hi = 5
        i.n = 3
        i.fn = 2
        i.setup()
175     def f1(i, xlst):
        return sum([-10*exp**(-0.2 * sqrt(xlst[k]**2 + xlst[k+1]**2)) for k in xrange
        (i.n -1)])
        def f2(i, xlst):
            a = 0.8
            b = 3
180         return sum([abs(x)**a + 5*sin(x)**b for x in xlst])

'''ZDT1'''
class ZDT1(Model):
    def __init__(i):
185     i.lo = 0
        i.hi = 1
        i.n = 30
        i.fn = 2
        i.setup()
190     def f1(i, xlst):
        return xlst[0]
        def f2(i, xlst):
            return (1 + 9 * (sum(xlst[1:]))/(i.n-1))
            # def f2(i,xlst):
            #     g1 = i.g(xlst)
            #     return g1*(1-sqrt(xlst[0]/g1))

195 '''ZDT3'''
class ZDT3(Model):
    def __init__(i):
200     i.lo = 0
        i.hi = 1
        i.n = 30
        i.fn = 2
        i.setup()
205     def f1(i, xlst):
        return xlst[0]
        def g(i, xlst):
            return (1 + (9/(i.n-1)) * sum(xlst[1:]))
210     def h(i,f1,g):
        return (1 - sqrt(f1/g) - f1/g) * sin(10 * pi * f1)
        def f2(i, xlst):
            return i.g(xlst) * i.h(i.f1(xlst),i.g(xlst))

215 '''Viennet3'''

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 4/5

```

class Viennet3(Model):
    def __init__(i):
        i.lo = -3
        i.hi = 3
220     i.n = 2
        i.fn = 3
        i.setup1()
        def setup1(i):
            i.min = 10**(5)
225     i.max = -10**(5)
            i.xy = Options(x = [i.generate_x()], y = [i.f1, i.f2, i.f3])
            i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range
            (i.fn)]) # hardcode 2
            i.history = {} # hold all logs for eras
        def f1(i, xlst):
230         xy2 = xlst[0]**2 + xlst[1]**2
            return 0.5* (xy2) + sin(xy2)
        def f2(i, xlst):
            x = xlst[0]
            y = xlst[1]
235         return ((3*x -2*y +4)**2/8 + (x-y+1)**2/27 + 15)
        def f3(i, xlst):
            xy2 = xlst[0]**2 + xlst[1]**2
            return (1/(xy2+1) - 1.1* exp**(-xy2))

240 '''DTLZ7'''
class DTLZ7(Model):
    def __init__(i):
        i.M = 20
        i.K = 20
245     i.lo = 0
        i.hi = 1
        i.n = i.M + i.K -1
        i.fn = i.M
        i.setup()
250     def fi(i, x): # the frist one is x[0]
        return x
        def fm(i, xh=0):
            return (1 + i.g())*i.h()
        def g(i):
255         return 1 + (9/i.K) * sum(i.xy.x[:i.M-1])
        def h(i):
            sumtemp = 0
            for n,x in enumerate(i.xy.x):
                if n ==i.M-2:
260                 break
                sumtemp +=(i.xy.y[n](x)/(1.0+i.g()))*(1+sin(3.0*pi*i.xy.y[n](x)))
            return (i.M - sumtemp)# k = 0,..., M-2
        def setup(i):
            tempx = i.generate_x()
            tempy = [i.fi for k in tempx[:-1]]
265         tempy.append(i.fm)
            i.xy = Options(x = tempx, y = tempy)
            i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range
            (i.fn)])
            i.history = {} # hold all logs for eras
270         i.min = 10**(5)
            i.max = -10**(5)
        def getDepen(i, xlst):
            temp = i.fm()
            return sum(xlst[:i.M])+temp

275 '''Schwefel's'''
class Schwefel(Model):
    def __init__(i):
        i.lo = -pi
        i.hi = pi
280     i.n = [10,20,40][0]
        i.f_bias = -460
        i.fn = 1
        i.randI = lambda x: random.randint(-x, x)
        i.randF = lambda x: random.uniform(-x, x)
285     i.a = [[i.randI(100) for _ in xrange(i.n)] for _ in xrange(i.n)] # matrix for a

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 5/5

```

i.b = [[i.randI(100) for _ in xrange(i.n)] for _ in xrange(i.n)] # matrix for b
i.alpha = [i.randF(pi) for _ in xrange(i.n)] # alpha
i.setup()
290 def f(i, x):
    F = sum([(i.A(n) - i.B(x,n))*2 for n in xrange(i.n)]) + i.f_bias
    return F
def A(i,n):
    sumA = sum([i.a[n][j]*sin(i.alpha[j]) + i.b[n][j] * cos(i.alpha[j]) for j in xrange(i.n)])
295     return sumA
def B(i, x,n):
    sumB = sum([i.a[n][j]*sin(s) + i.b[n][j]* cos(s) for j,s in enumerate(x)])
    return sumB
def setup(i):
300     i.min = 10**(5)
    i.max = -10**(5)
    i.xy = Options(x = [i.generate_x()], y = [i.f])
    i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range(i.fn)])
    i.history = { } # hold all logs for eras
305

```


Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 1/2

```

from __future__ import division
import sys, random, math
from base import *
from a12 import *
5 sys.dont_write_bytecode = True

''' All these are based on Dr.Menzies' tricks ^ sample codes'''

10 class Log():
    def __init__(i, tolog = []):
        i._cache, i.n, i._report = [], 0, None
        i.setup()
        map(i.__iadd__, tolog)
15 def __iadd__(i, tolog):
    if tolog == None: return tolog
    i.n += 1
    updated = False
    if len(i._cache) < Settings.other.keep:
20         i._cache += [tolog]
        updated = True
    else:
        if rand() <= Settings.other.keep/i.n:
            i._cache[int(rand()*Settings.other.keep)] = tolog
25         updated = True
    if updated:
        i._report = None
        i.updateLoHi(tolog)
        return i
30 def has(i):
    if i._report == None:
        i._report = i.report()
        return i._report

35 class Num(Log):
    def setup(i):
        i.lo = 10**5
        i.hi = -10**5
    def updateLoHi(i,x):
40         i.lo = min(i.lo, x)
        i.hi = max(i.hi, x)
    def median(i):
        n = len(i._cache)
        p = n//2
45         if (n % 2) : return i._cache[p]
        q = p + 1
        q = max(0, min(q,n))
        return (i._cache[p] + i._cache[q])/2
    def better(new,old):
50         "better if (1)less median or (2)same and less iqr"
        t = Settings.other.a12
        betterIqr = new.has().iqr < old.has().iqr
        new.lessp = False
        if new.lessp:
55             betterMed = new.has().median >= old.has().median
            same = a12(old._cache, new._cache) <= t
        else:
            betterMed = new.has().median <= old.has().median
            same = a12(new._cache, old._cache) <= t
60         return betterMed, same, betterIqr
    def report(i):
        sortedCache = sorted(i._cache)
        n = len(sortedCache)
        return Options(
65             median = i.median(),
            iqr = sortedCache[int(n*0.75)-int(n*0.5)],
            lo = i.lo,
            hi = i.hi)

70 @demo
def demoNum():
    for size in [16,32, 64,128, 256]:
        Settings.other.keep = size

```

Oct 14, 14 10:09

csc791sbse:hw7:Fu

Page 2/2

```

log = Num()
75 for x in xrange(100000): log +=x
    print size, ":", log.has().median

if __name__ == "__main__": eval(cmd())

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 1/3

```

from __future__ import division
import sys, random, math, datetime, time, re, pdb
from xtile import *
sys.dont_write_bytecode = True

5 rand= random.random

class Options: # "Thanks for Peter Norvig's trick"
10 def __init__(i, **d): i.__dict__.update(d)

Settings = Options(sa = Options(kmax = 1000,
                                score = {},
                                cooling = 0.6),
15 mws = Options(threshold = 0.0001,
                max_tries = 20,
                max_changes = 1000,
                prob = 0.25,
                score = {}
20 ),
                ga = Options(pop = 50,
                            crossRate = 0.6,
                            crossPoints = 2,
                            genNum = [100, 200, 400, 800]
25 ),
                de = Options(np= 100,
                            repeats = 100,
                            f = 0.75,
                            cr = 0.3,
                            threshold = 0.000001
30 ),
                pso = Options(N = 30,
                             w = 1,
                             phil = 2.8,
                             phi2 = 1.3,
                             threshold = 0.000001,
                             repeats = 1000,
40 ),
                other = Options(keep = 64,
                               baseline = 1000,
                               era = 50,
                               lives = 1,
                               show = False,
                               xtile = False,
45 a12 = [0.56, 0.64, 0.71][0],
                               repeats = 30,
                               reportrange = False))

def atom(x):
    try : return int(x)
    except ValueError:
        try : return float(x)
        except ValueError : return x

def cmd(com="demo('-h')"):
55 "Convert command line to a function call."
    if len(sys.argv) < 2: return com
    def strp(x): return isinstance(x, basestring)
    def wrap(x): return "%s"%x if strp(x) else str(x)
    words = map(wrap, map(atom, sys.argv[2:]))
60 return sys.argv[1] + '(' + ','.join(words) + ')'

def demo(f=None, cache=[]):
    def doc(d):
        return '%'+d.__doc__ if d.__doc__ else ""
65 if f == '-h':
        print '# sample demos'
        for n,d in enumerate(cache):
            print '%3s' % (n+1), d.func_name, doc(d)
        elif f:
            cache.append(f);
        else:
            s='|'+ '*40 +' + '\n'
            for d in cache:

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 2/3

```

    print '\n==|', d.func_name, s, doc(d), d()
75 return f

def reseed():
    seed = 1
    return random.seed(seed)

80 def say(mark):
    sys.stdout.write(mark)
    sys.stdout.flush()

85 def printlook(f):
    def wrapper(*lst): # tricks from Dr. Menzies
        ShowDate = datetime.datetime.now().strftime
        print "\n###", f.__name__, "##" * 50
        print "#", ShowDate("%Y-%m-%d %H:%M:%S")
90 beginTime = time.time()
        x = f(*lst)
        endTime = time.time()
        print "\n" + ("-"*60)
        dump(Settings, f.__name__)
95 print "\n# Runtime: %.3fsecs" % (endTime-beginTime)
        return x # return the searcher name and the results
    return wrapper

def dump(d, searchname, lvl = 0): # tricks from Dr. Menzies
100 d = d if isinstance(d, dict) else d.__dict__
    callableKey, line, gap = [], "", " "*lvl
    for k in sorted(d.keys()):
        val = d[k]
        if isinstance(val, (dict, Options)):
            callableKey += [k]
105 else:
            # if callable(val):
            # val = val.__name__
            line += (" {0}:{1}".format(k, val))
110 print gap + line
        for k in callableKey:
            if k == searchname or k == "other":
                print gap + (" {0}:{1}".format(k, "options"))
                dump(d[k], lvl+1)
115

def printReport(m, history):
    for i, f in enumerate(m.log.y):
        print "\n<f%s" % i
        for era in sorted(history.keys()):
            # pdb.set_trace()
            log = history[era].log.y[i]
            print str(era).rjust(7), xtile(log._cache, width = 33, show = "%5.2f", lo
120 = 0, hi = 1)

125 def printSumReport(m, history):
    # for i, f in enumerate(m.log.y):
    print "\n Objective Value"
    for era in sorted(history.keys()):
        # pdb.set_trace()
        log = [history[era].log.y[k] for k in range(len(m.log.y))]
130 ss = []
        ss.extend([log[s]._cache for s in range(len(log))])
        logsum = map(sum, zip(*ss))
        minvalue = min(logsum)
        maxvalue = max(logsum)
135 normlog = [(x - minvalue)/(maxvalue - minvalue + 0.00001) for x in logsum]
        print str(era).rjust(7), xtile(normlog, width = 33, show = "%5.2f", lo = 0,
        hi = 1)

def printRange(m, history):
140 rrange = {}
    # print sorted(m.history.keys())
    for i, f in enumerate(m.log.y):
        tlo=10**5
        thi=-10**5

```

Oct 14, 14 10:16

csc791sbse:hw7:Fu

Page 3/3

```
145     for era in sorted(history.keys()):
        # pdb.set_trace()
        if history[era].log.y[i].lo < tlo:
            tlo= history[era].log.y[i].lo
        if history[era].log.y[i].hi > tlo:
150            thi= history[era].log.y[i].hi
            temp = (round(tlo, 3), round(thi, 3))
            rrange[temp] =rrange.get(temp, 'f') +str(i) #{(0.0, 24.826): 'f0'}
    return  rrange
```