Tutorial section

Evolving polygons

Lunar explorer

Space probe(orbital, height, mass, boost velocity, initial velocity)

Examples section

I standard

II customized

III advanced

1. Standard Algorithm
2. Genetic algorithm
3. Evolution strategy
4. Simulated annealing
5. Differential evolution algorithm
6. Estimation of distribution algorithm
7. Pareto archived evolution strategy (PAES)
8. Non-dominated sorted genetic algorithm (NSGA II)
9. Particle swarm optimization
10. Ant colony optimization
11. Customized algorithm
12. Custom evolutionary computation
13. Custom archive
14. Custom observer
15. Custom replacer
16. Custom selector
17. Custom terminator
18. Custom variator

1. Advanced usage
2. Discrete optimization
3. Evaluating individuals concurrently
4. Island models
5. Replacement via Niching

Library reference

1. Library reference – Evolutionary computation
2. ec
3. emo
4. analysis
5. utilities
6. operators
7. archivers
8. evaluators
9. generators
10. migrators
11. observers
12. replacers
13. selectors
14. terminators
15. variators
16. Swarm intelligence
17. Swarm
18. topologies
19. Benchmark problems
20. Benchmarks
21. Single-objective Benchmark
22. Multi-objective Benchmark
23. Discrete Optimization Benchamrks
24. 1. ec-Evolutionary computation framework

A framework creating evolutionary computation

Class inspyred.ec.Bounder (lower, upper)

Example

Usage: Bounder(0,1) or ([0,0,0],[1,1,1]) => [0.1,-0.2,3] =>[0.1,0,1]

Meaning: evolutionary operators respect the legal bounds for candidates

Class inspyred.ec.DiscreteBounder(values)

Example

Usage: values=[1,4,8,16] [6,10,13,3] => [4,8,16,4]

Class inspyred.ec.Individual(candidate, maxmize)

Meaning: represent individual in an evolutionary computation

Usage: candidates owned fitness function generated fitness value ruled by maximize above

Class inspyred.ec.EvolutionaryComputation(random)

Encapsulates components of a genetic evolutionary computation

1. Selection mechanism
2. Variation operators
3. Replacement mechanism
4. Migration scheme
5. Archival mechanism
6. Teminators
7. Observers

Ps. Variator observer terminator can be specified as lists

(Pipeline) (sequence)(or)

General

Function evolve(generator,

Evaluator,

pop\_size=100,

seeds=None,

maximize=true,

bounder=None,

\*\*args)

Generator generate candidate

Evaluator evaluate candidate

Pop\_size number of individuals

Seeds iterable collection candidate in initial

Maximize Boolean of maximization

Bounder bound candidate

Args dictionary of keyword arguments

Below GA ES EDA DEA SA Inhereted Evolutionary computation

Class inspyred.ec.GA(random)

Key: rank selection; n-point crossover; bit-flip mutation

Optional parameter:

Number\_selecter: individuals number to be selected

Crossover\_rate: rate at crossover is performed

Num\_crossover\_points: crossover points

Mutation\_rate: mutation

Num\_elites: elites

Class inspyred.ec.ES(random)

Canonical evolution strategy

Key: selection, mutation, replacement with candidate solution of a sequence of real value

Class inspyred.ec.EDA(random)

Canonical estimation of distribution algorithm

Key: truncation selection, estimation of distribution variation, generation replacement, candidate solution is a sequence of real values

Class inspyred.ec.DEA(random)

Differential evolutionary algorithm

Key: tournament selection, heuristic crossover, Gaussian mutation, stedy-state replacement, candidate solution of real values

Class inspyred.ec.SA(random)

Simulated annealing

Key: selection, Gaussian mutation, simulated annealing replacement, candidate solution of real values

emo (Evolutionary multi-objective optimization)

Framework making multi objective evolutionary computation

Class inspyred.ec.emo.NSGA2

Non-dominated sorting genetic algorithm of Kalyanmoy Deb et al.

Key: non-dominated sorting with binary tournament selection, replacement, pareto archival strategy

Class inspyred.ec.emo.PAES(random)

Pareto archived evlution strategy of Joshua Knowles and David Corne

Key: (1+1)-ES, adaptive grid archive replacement

Class inspyred.ec.emo.Pareto(values=None, maximize=True)

Pareto multi-objective solution

Key: better rely on if it is better than or equal to the other solution in all objectives and strictly better in at least one objective

I3.analysis ---optimization result analysis

Analysis methods for the results of evolutionary computations

Inspyred.ec.analysis.allele\_plot(file, normalize=false,alleles=None,generation=None)

(single) meaning: plot the alleles from each generation from the individuals file

Inspyred.ec.analysis.fitness\_statistics(population)

Basic statistics of the populations fitness values

Inspyred.ec.analysis.generation\_plot(file,errorbars=true)

(single)basic statistics of the population’s fitness values

Inspyred.ec.analysis.hypervolume(pareto\_set,reference\_point=None)

Calculates hypervolume by slicing objectives (HSO)

[resource]( )

I4. Utilities ------optimization utility functions

Class inspyred.ec.utilities.objectify(func)

Key: provide each object own set of independent attributes

Inpyred.ec.utilities.memoize(func=None, maxlen=None)

Cache a function’s return value each time’s called instead of being re-evaluated.

I5. Operators

1. Arcchiver store separate solution
2. Evaluator fitness value
3. Generator generate new candidates
4. Migratory individual migration
5. Observers view progress
6. Replacer survivors of generation
7. Selector parents of generation
8. Terminators termination
9. Variators modify candidate
10. Archivers

General arguments:

1. Random
2. Population
3. Archive
4. Args

Inspyred.ec.archivers.adaptive\_grid\_archiver(1,2,3,4)

Meaning: best individuals, fixed grid

Typically for Pareto archived evolution strategy (PAES)

Inspyred.ec.archivers.best\_archiver(1,2,3,4)

Meaning: best, remove inferior

Typically for pareto archive

Inspyred.ec.archivers.default\_archiver(1,2,3,4)

Do nothing

Default return existing archive

Inspyred.ec.archivers.population\_archivers(1,2,3,4)

Meaning: replace archive with pop

1. Evaluator

General arguments:

1. Candidates
2. Args

Inspyred.ec.evaluators.evaluator(evaluate)

Evaluate inside contain fitness function

Inspyred.ec.evaluators.parallel\_evaluation\_mp(1,2)

Multiprocessing evaluation

Inspyred.ec.evaluators.parallel\_evaluation\_pp(1,2)

Parallel python evaluation

1. Generators

Create initial set of candidates

General arguments

1. Random
2. Args

Class inspyred.ec.generators.diversify(generator)

Ensure uniqueness

Inspyred.ec.generaors.Strategize(generator)

Extend candidate with strategy parameters

1. migrators

return uploaded population

general arguments

1. random
2. population
3. args

typically for island model evolutionary computation

class inspyred.ec.migrators.MultiprocessingMigrator(Max\_migrants=1)

multiprocessing migration

inspyred.ec.migrators.default\_migration(1,2,3)

do nothing return existing pop

1. 5. Observers

General arguments:

Population

Num\_generations

Num\_evaluations

Args

Class inspyred.ec.observers.emailobserver

Inspyred.ec.observers.archive\_observer

Inspyred.ec.observers.best\_observer

Inspyred.ec.observers.default\_observer

Inspyred.ec.observers.file\_observer

Inspyred.ec.observers.plot\_observer

Inspyred.ec.observers.population\_observer

Inspyred.ec.observers.stats\_observer

1. 6. Replacers

Survivor mechanism

General arguments:

1. Random
2. Population
3. Parents
4. Offspring
5. Args

Return surviving individual list

Prefix: inspyred.ec.replacers

Comma\_replacements(1,2,3,4,5)

‘comma’ replacement

Key: replaced size of the offspring at least large as the original population

Crowding\_replacement(1,2,3,4,5)

Crowding replacement

Key: closest individual to the current offspring replaced by the offspring

Default\_replacement(1,2,3,4,5)

Key: origin population

Generational\_replacement(1,2,3,4,5)

Key: offspring truncating to the pop size if larger

Nsga\_replacement(1,2,3,4,5)

Key: replaces population using non-dominated sorting technique from NSGA-II

Paes\_replacement(1,2,3,4,5)

Key: replaces using Pareto Archived evolution strategy method

Plus\_replacement(1,2,3,4,5)

Replaces by the best population many elements from the combined set of parents and offspring

Random\_replacement(1,2,3,4,5)

Key: replace random number of the pop

Simulated\_annealing\_replacement(1,2,3,4,5)

Key: simulated annealing schedule

Steady\_state\_replacement(1,2,3,4,5)

Key: keep at least individuals in the existing population

Truncation\_replacement(1,2,3,4,5)

Key: Best replaced from current population and offspring

1. 7.
2. Selectors

Parent selection return list of selected individuals

General arguments: 1. Random

2.Population

3.args

Prefix: inspyred.ec.selectors

Default\_selection(1,2,3)

Return population as selected

Fitness\_proportionate\_selection(1,2,3)

Stochastically chooses fitness probability based proportional individuals

Return fitness proportionate sampling

Rank\_selection(1,2,3)

Raw fitness determine rank choose individuals

Return a rank based sampling

Tournament\_selection(1,2,3)

Return tournament sampling

Random sample, tournament to best a selected

Truncation\_selection(1,2,3)

Return best individuals

Uniform\_selection(1,2,3)

Uniform selection by randomly choosing

1. terminators

return Boolean value true for ending

general arguments:

1. population
2. num\_generations
3. num\_evaluations
4. args

prefix: inspyred.ec.terminators

average\_fitness\_termination(1,2,3,4)

when average fitness near best fitness

default\_termination(1,2,3,4)

default always return true

diversity\_termination(1,2,3,4)

when population diversity less minimum

evaluation\_termination(1,2,3,4)

evaluation fitness meets or exceeds a maximum

generation\_termination(1,2,3,4)

number of generations meets or exceeds

no\_improvement\_termination(1,2,3,4)

best fitness value none change for a certain number of generations

time\_termination(1,2,3,4)

when elapsed time meet

user\_termination(1,2,3,4)

when user press key to

1. variators

return the list of modified individuals

general arguments:

1. random
2. candidates
3. args

crossover variators

pair of parents => a pair of offspring

mutation variators

candidate => single mutant

prefix: inspyred.ec.variators

default\_variation(1,2,3)

do nothing return the set of candidates

crossover(cross)

function decorator

sample: @crossover

def cross(random, mom, dad, args)

arithmetic\_crossover(random, mom, dad, args)

(AX) arithmetic crossover

Key: weight of allele

Return the offspring of arithmetic crossover

Blend\_crossover(random, mom, dad, args)

Blend crossover (BLX)

Key: AX plus A bit of mutation

Heuristic\_crossover(1,2,3)

Heuristic crossover(HX)

Used for particle swarm optimization required candidates can be pickled

Laplace\_crossover(random, mom, dad, args)

Laplace crossover(LX)

Deep and Thakur proposed crossover mutation

N\_point\_crossover(random, mom, dad, args)

n-point (NPX)

random cut and recombine

partially\_matched\_crossover(random, mom, dad, args)

partially matched crossover (PMX)

used for discrete values permutations

simulated\_binary\_crossover(random, mom, dad, args)

simulated binary crossover (SBX) with NSGA-II

cross\_over\_rate

Sbx\_distribution\_index down far ok

Up far not ok

Uniform\_crossover(random, mom, dad, args)

uniform crossover(UX)

biases coin flipped to determine offspring

mutator(mutate)

function decorator

example @ mutator

def mutate(1,2,3)

bit\_flip\_mutation(1,2,3)

bit\_flip mutation

key: bit-rate-flip

no bit – unchanged

Gaussian\_mutation(1,2,3)

Gaussian mutation

Key: mean, standard deviation, normal distribution

Inversion\_mutation(1,2,3)

Key: random location -> reverse sliced value

Nonuniform\_mutation(1,2,3)

Key: nonuniform mutation specified in Michalewicz ”GA+DS=Evolution Program” 1996

Random\_reset\_mutation(1,2,3)

Key: randomly choosing new values

Scramble\_mutation(1,2,3)

Key: randomly location -> scramble the sliced value

1. Swarm Intelligence

Class inspyred.swarm.ACS (random, components)

Ant Colony system discrete optimization

Class inspyred.swarm.PSO(random)

Basic particle swarm optimization algorithm

Deb and Padhge proposed

Class inspyred.swarm.TrailComponent(element, value, maximize=True, delta=1,epsilon=1)

Used as a discrete component of a trail of in ant colony optimization

Topologies ---- swarm topologies

Return list of lists of neighbors

Swarm intelligence -- particle swarms

Make use of topologies to determine relationship

Inspyred.swarm.topologies.ring\_topology(random,population,args)

Key: ring topology – all particles in a specified sized neighborhoods

Inspyred.swarm.topologies.star\_topology(random,population,args)

Key: star topology --- all particles as neighbors for all other particles

1. Benchmark Problems
2. Benchmarks

Benchmark optimization function

1. Single objective benchmarks
2. Multi-objective benchmarks
3. Discrete optimization benchmarks
4. benchmarks

Class inspyred.benchmark(dimensions, objectives = 1)

Abstract class define global optimization problem

Generator(candidates, args)

Evaluator(random, args)

Public attribute

Dimensions; objectives; bounder; maximize

Class inspyred.benchmarks.Binary(benchmark, dimension\_bits)

Existing benchmark problem

Represent by binary

1. single-objective benchmarks

class inspyred.benchmarks.Ackley(dimensions=2)

Ackley benchmark problem (global optimization problem)

Class inspyred.benchamrks.Griewank(dimensions = 2)

Griewank benchmark problem (gop)

Class inspyred.benchmarks.Rastrigin(dimensions = 2)

Rastrigin benchmark

Class inspyred.benchmarks.Rosebrock(dimensions = 2)

Rosenbrock benchmark

Class inspyred.benchmarks.Schwefel(dimensions=2)

Schwefel benchmark

Class inspyred.benchamrks.Sphere(dimensions = 2)

Sphere benchmark

1. multi-objective benchmarks

class inspyred.benchmarks.Kursawe(dimensions = 2)

kursawe multiobjective benchmark

key: n-dimensions to two dimensions

class inspyred.benchmarks.DTLZ1-2-3-4-5-6-7

N-dimensional inputs to m-dimensional outputs

1. discrete optimization benchmarks

class inspyred.benchmarks.Knapsack(capacity, items, duplicates = false)

knapsack benchmark

key: find the set of maximal value

that fit within a knapsack of

fixed weight capacity

class inspyred.benchmarks.TSP(weights)

traveling salesman benchmark

key: find the shortest visit route