Gnowee

Generated by Doxygen 1.8.6

Wed May 10 2017 20:42:18

Contents

Chapter 1

Main Page

Gnowee

Version

1.0

Gnowee is a general nearly-global metaheuristic optimization algorithm. It uses a blend of common heuristics to solve difficult gradient free constrained MINLP problems with mixed variables. It is capable of solving simpler problems, but may not be the algorithm of choice.

Running Gnowee

For examples on how to run Gnowee, please refer to the runGnowee notebook included in the src directory.

Building Documentation

To build the documentation, in the docs/src directory run the command:

$>>$ doxygen Doxyfile

This will build the html and latex version of the documentation. The symlink in the docs directory for the html index should automatically update. T

The up-to-date latex documentation is included in a pdf form in the repo under the docs directory. If an update of the latex documentation is desired, go to the docs/latex directory and run the command:

$>>$ make

This will build the latex documentation. The file will be named refman.pdf and be placed in this directory.

Citation Information

To cite Gnowee, use the following reference:

Contact information

Bugs and suggestions for improvement can be submitted via the GitHub page: `https://github.com/-SlaybaughLab/Gnowee`

Alternatively, questions or comments on Gnowee can be directed to:

James Bevins

james.e.bevins@gmail.com

Licensing Information

Acknowledements

AF, advisor, NSF

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 Constraints

Defines a class to perform constraint calculations.

Classes

- class Constraints.Constraint

  The class creates a Constraints object that can be used in optimization algorithms.

7.1.1 Detailed Description

Defines a class to perform constraint calculations.

Author

James Bevins,

Date

9May17

7.2 Gnowee

Main program for the Gnowee metaheuristic algorithm.

Functions

- def Gnowee.main

  Main controller program for the Gnowee optimization.

7.2.1 Detailed Description

Main program for the Gnowee metaheuristic algorithm. General nearly-global metaheuristic optimization algorithm. Uses a blend of common heuristics to solve difficult gradient free constrained MINLP problems with categorical variables. It is capable of solving simpler problems, but may not be the algorithm of choice.

For examples on how to run Gnowee, please refer to the runGnowee notebook included in the src directory.

Author

James Bevins

Date

9May17

7.2.2 Function Documentation

7.2.2.1 def Gnowee.main (*func*, *lb*, *ub*, *varType*, *gh*, *discreteVals* = [])

Main controller program for the Gnowee optimization.

Parameters

func	*function* The objective function to be minimized.
lb	*list or array* The lower bounds of the design variable(s). Only enter the bounds for continuous and integer/binary variables. The order must match the order specified in varType and ub.
ub	*list or array* The upper bounds of the design variable(s). Only enter the bounds for continuous and integer/binary variables. The order must match the order specified in varType and lb.
varType	*list or array* The type of variable for each position in the upper and lower bounds array. Discrete variables are to be included last as they are specified separatly from the lb/ub throught the discreteVals optional input. A variable can have two types (for example, 'dx' could denote a layer that can take multiple materials and be placed at multiple design locations) Allowed values: 'c' = continuous over a given range (range specified in lb & ub). 'i' = integer/binary (difference denoted by ub/lb). 'd' = discrete where the allowed values are given by the option discreteVals nxm array with n=# of discrete variables and m=# of values that can be taken for each variable. 'x' = combinatorial. All of the variables denoted by x are assumed to be "swappable" in combinatorial permutations. There must be at least two variables denoted as combinatorial. 'f' = fixed design variable. Will not be considered of any permutation.
gh	*GnoweeHeuristic object* An object constaining the settings and methods required for the Gnowee optimization algorithm.
discreteVals	*list of list(s)* nxm with n=# of discrete variables and m=# of values that can be taken for each variable. For example, if you had two variables representing the tickness and diameter of a cylinder that take standard values, the discreteVals could be specified as: discreteVals = [[0.125, 0.25, 0.375], [0.25, 0.5, 075]] Gnowee will then map the optimization results to these allowed values.

Returns

list: List for design event objects for the current top solution vs generation. Only stores the information when new optimal designs are found.

7.3 GnoweeHeuristics

Heuristics and settings supporting the Gnowee metaheuristic optimization algorithm.

Classes

- class GnoweeHeuristics.GnoweeHeuristics

  The class is the foundation of the Gnowee optimization algorithm.

Functions

- def GnoweeHeuristics.simple_bounds

  Application of problem boundaries to generated solutions.
- def GnoweeHeuristics.rejection_bounds

  Application of problem boundaries to generated solutions.

7.3.1 Detailed Description

Heuristics and settings supporting the Gnowee metaheuristic optimization algorithm. This instantiates the class and methods necessary to perform an optimization using the Gnowee algorithm. Each of the heuristics can also be used independently of the Gnowee algorithm by instantiating this class and choosing the desired heuristic.

The default settings are those found to be best for a suite of benchmark problems but one may find alternative settings are useful for the problem of interest based on the fitness landscape and type of variables.

Author

James Bevins

Date

8May17

7.3.2 Function Documentation

7.3.2.1 def GnoweeHeuristics.rejection_bounds (*parent, child, stepSize, lb, ub*)

Application of problem boundaries to generated solutions.

Adjusts step size for all rejected solutions until within the boundaries.

Parameters

parent	*array* The current system designs.
child	*array* The proposed new system designs.
stepSize	*float* The stepsize for the permutation.

lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).

Returns

array: The new system design that is within problem boundaries.

7.3.2.2 def GnoweeHeuristics.simple_bounds (*child, lb, ub*)

Application of problem boundaries to generated solutions.

If outside of the boundaries, the variable defaults to the boundary.

Parameters

child	*array* The proposed new system designs.
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).

Returns

array: The new system design that is within problem boundaries.

7.4 GnoweeUtilities

Classes and methods to support the Gnowee optimization algorithm.

Classes

- class GnoweeUtilities.Parent

  The class contains all of the parameters pertinent to a member of the population.
- class GnoweeUtilities.Event

  Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback.
- class GnoweeUtilities.ProblemParameters

  Creates an object containing key features of the chosen optimization problem.
- class GnoweeUtilities.Switch

  Creates a switch class object to switch between cases.

7.4.1 Detailed Description

Classes and methods to support the Gnowee optimization algorithm.

Author

James Bevins

Date

9May17

7.5 ObjectiveFunction

Defines a class to perform objective function calculations.

Classes

- class ObjectiveFunction.ObjectiveFunction

  The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Functions

- def ObjectiveFunction.prod

  Computes the product of a set of numbers (ie big PI, mulitplicative equivalent to sum).

7.5.1 Detailed Description

Defines a class to perform objective function calculations. This class contains the necessary functions and methods to create objective functions and initialize the necessary parameters. The class is pre-stocked with common benchmark functions for easy fishing.

Users can modify the this class to add additional functions following the format of the functions currently in the class.

Author

James Bevins

Date

10May17

7.5.2 Function Documentation

7.5.2.1 def ObjectiveFunction.prod (*iterable*)

Computes the product of a set of numbers (ie big PI, mulitplicative equivalent to sum).

Parameters

iterable	*list or array or generator* Iterable set to multiply.

Returns

float: The product of all of the items in iterable

7.6 OptiPlot

Plotting functions developed to help visualize and quantify the metaheuristic optimization process.

Functions

- def OptiPlot.plot_vars

    Plot the variables as they change in the optimization process.
- def OptiPlot.plot_hist

    Plots the histogram of function evaluation results from multiple runs of an optimization algorithm.
- def OptiPlot.plot_hist_comp

    Histograms and plots the comparison of two sets of function evaluation data.
- def OptiPlot.plot_feval_hist

    Plots the fitness vs function evaluation results of an optimization algorithm run.
- def OptiPlot.plot_tlf

    Plots a comparison of the TLF to the Levy distribution.
- def OptiPlot.plot_optimization

    Plots the results of optimization process for a given algorithm and parameter.

7.6.1 Detailed Description

Plotting functions developed to help visualize and quantify the metaheuristic optimization process.

Author

James Bevins

Date

10May17

7.6.2 Function Documentation

7.6.2.1 def OptiPlot.plot_feval_hist (*data* = [ ], *listData* = [ ], *label* = [ ])

Plots the fitness vs function evaluation results of an optimization algorithm run.

Can plot a single run or multiple to compare results. To plot multiple data sets, use the listData argument; otherwise, use the data argument.

Parameters

data	*list or array* Contains the function eval history. Columns are: [function evals, fitness, number of datapoints].
listData	*list of lists or arrays* Contains a list of function eval histories. Columns are: [function evals, fitness, number of datapoints].

label	*list* List of names corresponding to the data sets provided.

7.6.2.2 def OptiPlot.plot_hist (*data, title = '', xLabel = ''*)

Plots the histogram of function evaluation results from multiple runs of an optimization algorithm.

Can be used to understand the convergence of the algorithm.

Parameters

data	*list* Contains the number of function evals for each optimization run.
title	*string* Title for plot.
xLabel	*string* Label for independent variable.

7.6.2.3 def OptiPlot.plot_hist_comp (*data, data2, dataLabels, title = '', xLabel = ''*)

Histograms and plots the comparison of two sets of function evaluation data.

Parameters

data	*list* Contains the number of function evals for each optimization run.
data2	*list* Contains the number of function evals for each optimization run for a second set of runs.
dataLabels	*list* Contains the legend label names for each data set.
title	*string* Title for plot.
xLabel	*string* Label for independent variable.

7.6.2.4 def OptiPlot.plot_optimization (*data, label, title = ''*)

Plots the results of optimization process for a given algorithm and parameter.

Parameters

data	*array* Contains the function eval history. Columns are: [function evals, fitness, number of datapoints]

label	*list* List of names of the problem types ran.
title	*string* Title for plot.

7.6.2.5 def OptiPlot.plot_tlf (*alpha* = `1.5`, *gamma* = `1.`, *numSamp* = `1E7`, *cutPoint* = `10.`)

Plots a comparison of the TLF to the Levy distribution.

Parameters

alpha	*float* Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
gamma	*float* Gamma - Scale unit of process for Levy flights.
numSamp	*integer* Number of Levy flights to sample.
cutPoint	*float* Point at which to cut sampled Levy values and resample.

7.6.2.6 def OptiPlot.plot_vars (*data*, *lowBounds* = `[]`, *upBounds* = `[]`, *title* = `[]`, *label* = `[]`)

Plot the variables as they change in the optimization process.

Currently only functions in post-processing, not real time.

Parameters

data	*list of event objects* Contain the optimization history in event objects within the data list.
lowBounds	*array* The lower bounds of the design variable(s).
upBounds	*array* The upper bounds of the design variable(s).
title	*string* Title for plot.
label	list List of names of design variables.

7.7 Sampling

Different methods to perform phase space sampling and random walks.

Functions

- def Sampling.initial_samples

  Generate a set of samples in a given phase space.
- def Sampling.plot_samples

  Plot the first 2 and 3 dimensions on the sample distribution.
- def Sampling.levy

  Sample the Levy distribution given by.
- def Sampling.tlf

  Samples from a truncated Levy flight distribution (TLF) according to Manegna, "Stochastic Process with Ultraslow Convergence to a Gaussian: The Truncated Levy Flight" to map a levy distribution onto the interval [0,1].
- def Sampling.NOLH

  This library allows to generate Nearly Orthogonal Latin Hypercubes (NOLH) according to Cioppa (2007) and De Rainville et al.
- def Sampling.params

  Returns the NOLH order m, the required configuration length q and the number of columns to remove to obtain the desired dimensionality.
- def Sampling.get_cdr_permutations

  Generate a set of CDR permulations for NOLH.

7.7.1 Detailed Description

Different methods to perform phase space sampling and random walks. Design of experiment and phase space sampling methods. Includes some vizualization tools.

Dependencies on pyDOE.

Author

James Bevins

Date

8May17

7.7.2 Function Documentation

7.7.2.1 def Sampling.get_cdr_permutations (*dim*)

Generate a set of CDR permulations for NOLH.

Parameters

dim	*integer* The dimension of the space.

Returns

array: A configuration vector.
array: Array containing the indexes of the colummns to be removed from conf vector.

7.7.2.2 def Sampling.initial_samples (*lb, ub, method, numSamp*)

Generate a set of samples in a given phase space.

The current methods available are 'random', 'nolh', 'nolh-rp', 'nolh-cdr', or 'lhc'.

Parameters

lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
method	*string* String representing the chosen sampling method. Valid options are: 'random', 'nolh', 'nolh-rp', 'nolh-cdr', or 'lhc'.
numSamp	*integer* The number of samples to be generated. Ignored for nolh algorithms.

Returns

array: The list of coordinates for the sampled phase space.

7.7.2.3 def Sampling.levy (*nc, nr* = 0, *alpha* = 1.5, *gam* = 1, *n* = 1)

Sample the Levy distribution given by.

$L_{\alpha,\gamma}(z) = \frac{1}{\pi} \int\limits_{0}^{+\infty} e^{-\gamma q^{\alpha}} \cos(qz) dq$

using the Mantegna algoritm outlined in "Fast, Accurate Algorithm for Numerical Simulation of Levy Stable Stochastic Processes."

Parameters

nc	*integer* The number of columns of Levy values for the return array.
nr	*integer* The number of rows of Levy values for the return array.
alpha	*float* Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
gam	*float* Gamma - Scale unit of process for Levy flights.
n	*integer* Number of independent variables - can be used to reduce Levy flight sampling variance.

Returns

array: Array representing the levy flights for each nest.

7.7.2.4 def Sampling.NOLH (*conf*, *remove* = `None`)

This library allows to generate Nearly Orthogonal Latin Hypercubes (NOLH) according to Cioppa (2007) and De Rainville et al.

(2012) and reference therein.

https://pypi.python.org/pypi/pynolh

Constructs a Nearly Orthogonal Latin Hypercube (NOLH) of order *m* from a configuration vector *conf*. The configuration vector may contain either the numbers in $ [0 q-1] $ or $ [1 q] $ where $ q = 2^{m-1} $. The columns to be *removed* are also in $ [0 d-1] $ or $ [1 d] $ where

$ d = m + {m-1}{2} $

is the NOLH dimensionality.

The whole library is incorporated here with minimal modification for commonality and consolidation of methods.

Parameters

conf	*array* Configuration vector.
remove	*array* Array containing the indexes of the colummns to be removed from conf vector.

Returns

array: Array containing nearly orthogonal latin hypercube sampling.

7.7.2.5 def Sampling.params (*dim*)

Returns the NOLH order m, the required configuration length q and the number of columns to remove to obtain the desired dimensionality.

Parameters

dim	*integer* The dimension of the space.

7.7.2.6 def Sampling.plot_samples (*s*)

Plot the first 2 and 3 dimensions on the sample distribution.

Can't plot the full hyperspace yet. Produces a very simple plot for visualizing the difference in the sampling methods.

Parameters

s	*array* The list of coordinates for the sampled phase space.

7.7.2.7 def Sampling.tlf (*numRow* = `1`, *numCol* = `1`, *alpha* = `1.5`, *gam* = `1.`, *cutPoint* = `10.`)

Samples from a truncated Levy flight distribution (TLF) according to Manegna, "Stochastic Process with Ultraslow Convergence to a Gaussian: The Truncated Levy Flight" to map a levy distribution onto the interval [0,1].

Parameters

numRow	*integer* Number of rows of Levy flights to sample.
numCol	*integer* Number of columns of Levy flights to sample.
alpha	*float* Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
gam	*float* Gamma - Scale unit of process for Levy flights.
cutPoint	*float* Point at which to cut sampled Levy values and resample.

Returns

array: Array representing the levy flights on the interval (0,1).

Chapter 8

Namespace Documentation

8.1 Gnowee Namespace Reference

Contains the Gnowee optimization program and associated utilities.

Functions

- def main

  Main controller program for the Gnowee optimization.

8.1.1 Detailed Description

Contains the Gnowee optimization program and associated utilities.

Version

1.0

General nearly-global metaheuristic optimization algorithm. Uses a blend of common heuristics to solve difficult gradient free constrained MINLP problems with categorical variables. It is capable of solving simpler problems, but may not be the algorithm of choice.

For examples on how to run Gnowee, please refer to the runGnowee notebook included in the src directory.

Author

James Bevins

Date

9May17

See Also

Gnowee
GnoweeHeuristics
GnoweeUtilities
ObjectiveFunction
Constraints
OptiPlot
Sampling

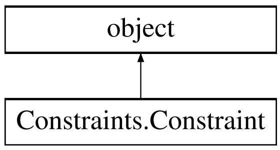
Chapter 9

Class Documentation

9.1 Constraints.Constraint Class Reference

The class creates a Constraints object that can be used in optimization algorithms.

Inheritance diagram for Constraints.Constraint:



Public Member Functions

- def __init__

  Constructor to build the ObjectiveFunction class.
- def __repr__

  Constraint class param print function.
- def __str__

  Human readable Constraint print function.
- def set_constraint_func

  Converts an input string name for a function to a function handle.
- def get_penalty

  Calculate the constraint violation penalty, if any.
- def less_or_equal

  Compares a previously calculated value to a user specifed maximum including that maximum.
- def less_than

  Compares a previously calculated value to a user specifed maximum excluding that maximum.
- def greater_than

  Compares the calculated value to the minimum specified by the user.

Public Attributes

- func

  function handle: The function handle for the constraint function to be used for the optimization.
- constraint

  float: The constraint to be enforced.

- penalty

  float: The penalty to be applied if the constraint is violated

9.1.1 Detailed Description

The class creates a Constraints object that can be used in optimization algorithms.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 def Constraints.Constraint.__init__ (*self*, *method* = `None`, *constraint* = `None`, *penalty* = `1E15`)

Constructor to build the ObjectiveFunction class.

Parameters

self	*object pointer* The object pointer.
method	*string* The name of the constraint function to evaluate.
constraint	*float* The constraint to be compared against.
penalty	*float* The penalty to be applied if a constraint is violated. 1E15 is recommended.

9.1.3 Member Function Documentation

9.1.3.1 def Constraints.Constraint.__repr__ (*self*)

Constraint class param print function.

Parameters

self	*pointer* The Constraint pointer.

9.1.3.2 def Constraints.Constraint.__str__ (*self*)

Human readable Constraint print function.

Parameters

self	*pointer* The Constraint pointer.

9.1.3.3 def Constraints.Constraint.get_penalty (*self*, *violation*)

Calculate the constraint violation penalty, if any.

Parameters

self	*pointer* The Constraint pointer.
violation	*float* The magnitude of the constraint violation used for scaling the penalty.

Returns

float: The scaled penalty.

9.1.3.4 def Constraints.Constraint.greater_than (*self, candidate*)

Compares the calculated value to the minimum specified by the user.

Parameters

self	*pointer* The Constraint pointer.
candidate	*float* The calculated value corresponding to a candidate design.

Returns

float: The penalty associated with the candidate design.

9.1.3.5 def Constraints.Constraint.less_or_equal (*self, candidate*)

Compares a previously calculated value to a user specifed maximum including that maximum.

Parameters

self	*pointer* The Constraint pointer.
candidate	*float* The calculated value corresponding to a candidate design.

Returns

float: The penalty associated with the candidate design.

9.1.3.6 def Constraints.Constraint.less_than (*self, candidate*)

Compares a previously calculated value to a user specifed maximum excluding that maximum.

Parameters

self	*pointer* The Constraint pointer.
candidate	*float* The calculated value corresponding to a candidate design.

Returns

float: The penalty associated with the candidate design.

9.1.3.7 def Constraints.Constraint.set_constraint_func (*self*, *funcName*)

Converts an input string name for a function to a function handle.

Parameters

self	*pointer* The Constraint pointer.
funcName	*string* A string identifying the constraint function to be used.

9.1.4 Member Data Documentation

9.1.4.1 Constraints.Constraint.constraint

float: The constraint to be enforced.

9.1.4.2 Constraints.Constraint.func

function handle: The function handle for the constraint function to be used for the optimization.

The function must be specified as a method of the class.

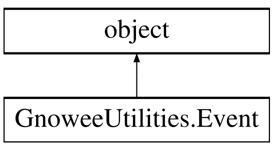
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Constraints.py

9.2 GnoweeUtilities.Event Class Reference

Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback.

Inheritance diagram for GnoweeUtilities.Event:



Public Member Functions

- def __init__

  Constructor to build the Event class.
- def __repr__

  Event print function.
- def __str__

  Human readable Event print function.

Public Attributes

- generation

  integer: The generation the design was arrived at.
- evaluations

  integer: The number of fitness evaluations done to obtain this design.
- fitness

  float: The assessed fitness for the current set of variables.
- design

  array: The set of variables representing a design solution.

9.2.1 Detailed Description

Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 def GnoweeUtilities.Event.__init__ (*self*, *generation*, *evaluations*, *fitness*, *design*)

Constructor to build the Event class.

Parameters

self	*Event pointer* The Event pointer.
generation	*integer* The generation the design was arrived at.
evaluations	*integer* The number of fitness evaluations done to obtain this design.
fitness	*float* The assessed fitness for the current set of variables.
design	*array* The set of variables representing a design solution.

9.2.3 Member Function Documentation

9.2.3.1 def GnoweeUtilities.Event.__repr__ (*self*)

Event print function.

Parameters

self	*Event pointer* The Event pointer.

9.2.3.2 def GnoweeUtilities.Event.__str__ (*self*)

Human readable Event print function.

Parameters

self	*Event pointer* The Event pointer.

9.2.4 Member Data Documentation

9.2.4.1 GnoweeUtilities.Event.design

array: The set of variables representing a design solution.

9.2.4.2 GnoweeUtilities.Event.evaluations

integer: The number of fitness evaluations done to obtain this design.

9.2.4.3 GnoweeUtilities.Event.fitness

float: The assessed fitness for the current set of variables.

9.2.4.4 GnoweeUtilities.Event.generation

integer: The generation the design was arrived at.

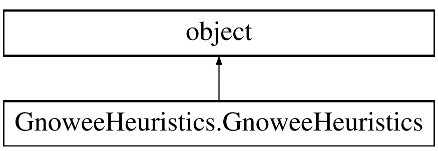
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py

9.3 GnoweeHeuristics.GnoweeHeuristics Class Reference

The class is the foundation of the Gnowee optimization algorithm.

Inheritance diagram for GnoweeHeuristics.GnoweeHeuristics:



Public Member Functions

- def __init__

  Constructor to build the GnoweeHeuristics class.
- def __repr__

  GnoweeHeuristics print function.
- def __str__

  Human readable GnoweeHeuristics print function.
- def initialize

  Initialize the population according to the sampling method chosen.
- def disc_levy_flight

  Generate new children using truncated Levy flights permutation of current generation design parameters according to:
- def cont_levy_flight

  Generate new children using Levy flights permutation of current generation design parameters according to:
- def scatter_search

  Generate new designs using the scatter search heuristic according to:
- def elite_crossover

  Generate new designs by using inver-over on combinatorial variables.
- def crossover

  Generate new children using distance based crossover strategies on the top parent.
- def mutate

  Generate new children by adding a weighted difference between two population vectors to a third vector.

Public Attributes

- population

  integer The number of members in each generation.
- initSampling

  string The method used to sample the phase space and create the initial population.
- fracDiscovered

  float Discovery probability used for the mutate() heuristic.
- fracElite

  float Elite fraction probability used for the scatter_search(), crossover(), and cont_crossover() heuristics.
- fracLevy

  float Levy flight probability used for the disc_levy_flight() and cont_levy_flight() heuristics.
- alpha

  float Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
- gamma

  float Gamma - scale unit of process for Levy flights.
- n

  integer Number of independent variables - can be used to reduce Levy flight sampling variance.
- scalingFactor

  float Step size scaling factor used to adjust Levy flights to length scale of system.
- penalty

  float Individual constraint violation penalty to add to objective function.
- maxGens

  integer The maximum number of generations to search.
- maxFevals

  integer The maximum number of objective function evaluations.
- convTol

float The minimum change of the best objective value before the search terminates.

- stallLimit

  integer The maximum number of gen3rations to search without a descrease exceeding convTol.
- optimalFitness

  float The best know fitness value for the problem considered used to test for convergence.
- optConvTol

  float The maximum deviation from the best know fitness value before the search terminates.

9.3.1 Detailed Description

The class is the foundation of the Gnowee optimization algorithm.

It sets the settings required for the algorithm and defines the heurstics.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 def GnoweeHeuristics.GnoweeHeuristics.__init__ (*self*, *population* = `25`, *initSampling* = `'lhc'`, *fracDiscovered* = `0.2`, *fracElite* = `0.2`, *fracLevy* = `0.2`, *alpha* = `1.5`, *gamma* = `1`, *n* = `1`, *scalingFactor* = `10.0`, *penalty* = `0.0`, *maxGens* = `20000`, *maxFevals* = `200000`, *convTol* = `1e-6`, *stallLimit* = `225`, *optimalFitness* = `0`, *optConvTol* = `1e-2`)

Constructor to build the GnoweeHeuristics class.

The default settings are found to be optimized for a wide range of problems, but can be changed to optimize performance for a particular problem type or class. For more details, refer to the benchmark code in the development branch of the repo or <insert link="" to="" paper>="">.

If the optimizal fitness is unknown, as it often is, this can be left as zero or some reasonable guess based on the understanding of the problem. If the opimtimal fitness is set below what is actually obatinable, the only impact is the removal of this convergence criteria, and the program will still run.

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
population	*integer* The number of members in each generation.
initSampling	*string* The method used to sample the phase space and create the initial population. Valid options are 'random', 'nolh', 'nolh-rp', 'nolh-cdr', and 'lhc' as specified in init_samples().
fracDiscovered	: *float* Discovery probability used for the mutate() heuristic.
fracElite	*float* Elite fraction probability used for the scatter_search(), crossover(), and cont_crossover() heuristics.

fracLevy	*float* Levy flight probability used for the disc_levy_flight() and cont_levy_flight() heuristics.
alpha	*float* Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
gamma	*float* Gamma - scale unit of process for Levy flights.
n	*integer* Number of independent variables - can be used to reduce Levy flight sampling variance.
penalty	*float* Individual constraint violation penalty to add to objective function.
scalingFactor	*float* Step size scaling factor used to adjust Levy flights to length scale of system. The implementation of the Levy flight sampling makes this largely arbitrary.
maxGens	*integer* The maximum number of generations to search.
maxFevals	*integer* The maximum number of objective function evaluations.
convTol	*float* The minimum change of the best objective value before the search terminates.
stallLimit	*integer* The maximum number of generations to search without a descrease exceeding convTol.
optimalFitness	*float* The best know fitness value for the problem considered used to test for convergence.
optConvTol	*float* The maximum deviation from the best know fitness value before the search terminates.

9.3.3 Member Function Documentation

9.3.3.1 def GnoweeHeuristics.GnoweeHeuristics.__repr__ (*self*)

GnoweeHeuristics print function.

Parameters

self	*GnoweeHeuristics pointer* The GnoweeHeuristics pointer.

9.3.3.2 def GnoweeHeuristics.GnoweeHeuristics.__str__ (*self*)

Human readable GnoweeHeuristics print function.

Parameters

self	*GnoweeHeuristics pointer* The GnoweeHeuristics pointer.

9.3.3.3 def GnoweeHeuristics.GnoweeHeuristics.cont_levy_flight (*self*, *pop*, *lb*, *ub*, *varID*)

Generate new children using Levy flights permutation of current generation design parameters according to:

$x_r^{g+1} = x_r^g + \frac{1}{\beta} L_{\alpha,\gamma}$,

where $L_{\alpha,\gamma}$ is calculated in levy() according to the Mantegna algorithm. Applies rejection_bounds() to ensure all solutions lie within the design space by adapting the step size to the size of the design space.

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
pop	*list of arrays* The current parent sets of design variables representing system designs for the population.
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varID	*array* A truth array indicating the location of the variables to be permuted. If the variable is to be permuted, a 1 is inserted at the variable location; otherwise a 0.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.
list: A list of the identities of the chosen index for each child.

9.3.3.4 def GnoweeHeuristics.GnoweeHeuristics.crossover (*self*, *pop*, *lb*, *ub*, *varID*, *intDiscID* = `None`)

Generate new children using distance based crossover strategies on the top parent.

Ideas adapted from Walton "Modified Cuckoo Search: A New Gradient Free Optimisation Algorithm" and Storn "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces"

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
pop	*list of arrays* The current parent sets of design variables representing system designs for the population.

lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varID	*array* A truth array indicating the location of the variables to be permuted. If the variable is to be permuted, a 1 is inserted at the variable location; otherwise a 0.
intDiscID	*array* A truth array indicating the location of the discrete variable. A 1 is inserted at the discrete variable location; otherwise a 0. If no discrete variables, the array will be set to 0 automatically.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.
list: A list of the identities of the chosen index for each child.

9.3.3.5 def GnoweeHeuristics.GnoweeHeuristics.disc_levy_flight (*self, pop, lb, ub, varID*)

Generate new children using truncated Levy flights permutation of current generation design parameters according to:

$L_{\alpha,\gamma} = FLOOR(TLF_{\alpha,\gamma} * D(x)),$

where $TLF_{\alpha,\gamma}$ is calculated in tlf(). Applies rejection_bounds() to ensure all solutions lie within the design space by adapting the step size to the size of the design space.

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
pop	*list of arrays* The current parent sets of design variables representing system designs for the population.
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varID	*array* A truth array indicating the location of the variables to be permuted. If the variable is to be permuted, a 1 is inserted at the variable location; otherwise a 0.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.
list: A list of the identities of the chosen index for each child.

9.3.3.6 def GnoweeHeuristics.GnoweeHeuristics.elite_crossover (*self, pop*)

Generate new designs by using inver-over on combinatorial variables.

Adapted from ideas in Tao, "Iver-over Operator for the TSP."

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
pop	*list of arrays* The current parent sets of design variables representing system designs for the population.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.

9.3.3.7 def GnoweeHeuristics.GnoweeHeuristics.initialize (*self, numSamples, sampleMethod, lb, ub, varType*)

Initialize the population according to the sampling method chosen.

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
numSamples	*integer* The number of samples to be generated.
sampleMethod	*string* The method used to sample the phase space and create the initial population. Valid options are 'random', 'nolh', 'nolh-rp', 'nolh-cdr', and 'lhc' as specified in init_samples().
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varType	*array* The type of variable for each design parameter. Allowed values: 'c' = continuous 'i' = integer/binary (difference denoted by ub/lb) 'd' = discrete where the allowed values are given by the option discreteVals nxm array with n=# of discrete variables and m=# of values that can be taken for each variable 'x' = combinatorial. All of the variables denoted by x are assumed to be "swappable" in combinatorial permutations. There must be at least two variables denoted as combinatorial. 'f' = fixed design variable

Returns

list of arrays: The initialized set of samples.

9.3.3.8 def GnoweeHeuristics.GnoweeHeuristics.mutate (*self, pop, lb, ub, varID, intDiscID* = `None`)

Generate new children by adding a weighted difference between two population vectors to a third vector.

Ideas adapted from Storn, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces" and Yang, "Nature Inspired Optimmization Algorithms"

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.
pop	*list of arrays* The current parent sets of design variables representing system designs for the population.
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varID	*array* A truth array indicating the location of the variables to be permuted. If the variable is to be permuted, a 1 is inserted at the variable location; otherwise a 0.
intDiscID	*array* A truth array indicating the location of the discrete variable. A 1 is inserted at the discrete variable location; otherwise a 0. If no discrete variables, the array will be set to 0 automatically.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.

9.3.3.9 def GnoweeHeuristics.GnoweeHeuristics.scatter_search (*self*, *pop*, *lb*, *ub*, *varID*, *intDiscID* = `None`)

Generate new designs using the scatter search heuristic according to:

$x^{g+1} = c_1 + (c_2 - c_1) r$

where

$c_1 = x^e - d(1 + \alpha\beta)$

$c_2 = x^e - d(1 - \alpha\beta)$

$d = \frac{x^r - x^e}{2}$

and

$\alpha =$ 1 if i $<$ j & -1 if i $>$ j

$\beta = \frac{|j-i|-1}{b-2}$

where b is the size of the population.

Adapted from ideas in Egea, "An evolutionary method for complex- process optimization."

Applies simple_bounds() to ensure all solutions lie within the design space by adapting the step size to the size of the design space.

Parameters

self	*GnoweeHeuristic pointer* The GnoweeHeuristics pointer.

pop	*list of arrays* The current parent sets of design variables representing system designs for the population.
lb	*array* The lower bounds of the design variable(s).
ub	*array* The upper bounds of the design variable(s).
varID	*array* A truth array indicating the location of the variables to be permuted. If the variable is to be permuted, a 1 is inserted at the variable location; otherwise a 0.
intDiscID	*array* A truth array indicating the location of the discrete variable. A 1 is inserted at the discrete variable location; otherwise a 0. If no discrete variables, the array will be set to 0 automatically.

Returns

list of arrays: The proposed children sets of design variables representing the updated design parameters.
list: A list of the identities of the chosen index for each child.

9.3.4 Member Data Documentation

9.3.4.1 GnoweeHeuristics.GnoweeHeuristics.alpha

float Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.

9.3.4.2 GnoweeHeuristics.GnoweeHeuristics.convTol

float The minimum change of the best objective value before the search terminates.

9.3.4.3 GnoweeHeuristics.GnoweeHeuristics.fracDiscovered

float Discovery probability used for the mutate() heuristic.

9.3.4.4 GnoweeHeuristics.GnoweeHeuristics.fracElite

float Elite fraction probability used for the scatter_search(), crossover(), and cont_crossover() heuristics.

9.3.4.5 GnoweeHeuristics.GnoweeHeuristics.fracLevy

float Levy flight probability used for the disc_levy_flight() and cont_levy_flight() heuristics.

9.3.4.6 GnoweeHeuristics.GnoweeHeuristics.gamma

float Gamma - scale unit of process for Levy flights.

9.3.4.7 GnoweeHeuristics.GnoweeHeuristics.initSampling

string The method used to sample the phase space and create the initial population.

Valid options are 'random', 'nolh', 'nolh-rp', 'nolh-cdr', and 'lhc' as specified in init_samples().

9.3.4.8 GnoweeHeuristics.GnoweeHeuristics.maxFevals

integer The maximum number of objective function evaluations.

9.3.4.9 GnoweeHeuristics.GnoweeHeuristics.maxGens

integer The maximum number of generations to search.

9.3.4.10 GnoweeHeuristics.GnoweeHeuristics.n

integer Number of independent variables - can be used to reduce Levy flight sampling variance.

9.3.4.11 GnoweeHeuristics.GnoweeHeuristics.optConvTol

float The maximum deviation from the best know fitness value before the search terminates.

9.3.4.12 GnoweeHeuristics.GnoweeHeuristics.optimalFitness

float The best know fitness value for the problem considered used to test for convergence.

9.3.4.13 GnoweeHeuristics.GnoweeHeuristics.penalty

float Individual constraint violation penalty to add to objective function.

9.3.4.14 GnoweeHeuristics.GnoweeHeuristics.population

integer The number of members in each generation.

9.3.4.15 GnoweeHeuristics.GnoweeHeuristics.scalingFactor

float Step size scaling factor used to adjust Levy flights to length scale of system.

The implementation of the Levy flight sampling makes this largely arbitrary.

9.3.4.16 GnoweeHeuristics.GnoweeHeuristics.stallLimit

integer The maximum number of gen3rations to search without a descrease exceeding convTol.

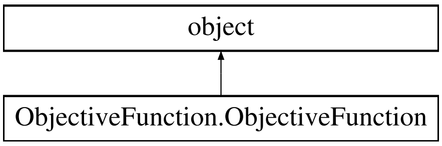
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeHeuristics.py

9.4 ObjectiveFunction.ObjectiveFunction Class Reference

The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Inheritance diagram for ObjectiveFunction.ObjectiveFunction:



Public Member Functions

- def __init__

  Constructor to build the ObjectiveFunction class.
- def __repr__

  ObjectiveFunction class param print function.
- def __str__

  Human readable ObjectiveFunction print function.
- def set_obj_func

  Converts an input string name for a function to a function handle.
- def spring

  Spring objective function with penalty method of constraint enforcement.
- def mi_spring

  Spring objective function with penalty method of constraint enforcement.
- def welded_beam

  Welded Beam objective function with penalty method of constraint enforcement.
- def pressure_vessel

  Pressure vessel objective function with penalty method of constraint enforcement.
- def mi_pressure_vessel

  Mixed Integer Pressure vessel objective function with penalty method of constraint enforcement.
- def speed_reducer

  Speed reducer objective function with penalty method of constraint enforcement.
- def mi_chemical_process

  Chemical process design mixed integer problem.
- def ackley

  Ackley Function: Mulitmodal, n dimensional.
- def shifted_ackley

  Ackley Function: Mulitmodal, n dimensional Ackley Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.
- def dejong

  De Jong Function: Unimodal, n-dimensional.
- def shifted_dejong

  De Jong Function: Unimodal, n-dimensional De Jong Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.
- def easom

  Easom Function: Multimodal, n-dimensional.
- def shifted_easom

  Easom Function: Multimodal, n-dimensional Easom Function that is shifted from the symmetric pi, pi optimimum.
- def griewank

  Griewank Function: Multimodal, n-dimensional.
- def shifted_griewank

  Griewank Function: Multimodal, n-dimensional Griewank Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.
- def rastrigin

  Rastrigin Function: Multimodal, n-dimensional.

- def shifted_rastrigin

  Rastrigin Function: Multimodal, n-dimensional Rastrigin Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.
- def rosenbrock

  Rosenbrock Function: uni-modal, n-dimensional.
- def shifted_rosenbrock

  Rosenbrock Function: uni-modal, n-dimensional Rosenbrock Function that is shifted from the symmetric 0,0,0...0 optimimum.
- def tsp

  Generic objective funtion to evaluate the TSP optimization by calculating total distance traveled.

Public Attributes

- func

  function handle The function handle for the objective function to be used for the optimization.
- objective

  integer, float, or numpy array The desired outcome of the optimization.

9.4.1 Detailed Description

The class creates a ObjectiveFunction object that can be used in optimization algorithms.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 def ObjectiveFunction.ObjectiveFunction.__init__ (*self*, *method* = `None`, *objective* = `None`)

Constructor to build the ObjectiveFunction class.

Parameters

self	*pointer* The ObjectiveFunction pointer.
method	*string* The name of the objective function to evaluate.
objective	*integer, float, or numpy array* The desired objective associated with the optimization. The chosen value and type must be compatible with the optiization function chosen.

9.4.3 Member Function Documentation

9.4.3.1 def ObjectiveFunction.ObjectiveFunction.__repr__ (*self*)

ObjectiveFunction class param print function.

Parameters

self	*pointer* The ObjectiveFunction pointer.

9.4.3.2 def ObjectiveFunction.ObjectiveFunction.__str__ (*self*)

Human readable ObjectiveFunction print function.

Parameters

self	*pointer* The ObjectiveFunction pointer.

9.4.3.3 def ObjectiveFunction.ObjectiveFunction.ackley (*self*, *u*, *penalty* = 0.0)

Ackley Function: Mulitmodal, n dimensional.

Optimal example:

u = [0, 0, 0, 0, ... n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.4 def ObjectiveFunction.ObjectiveFunction.dejong (*self*, *u*, *penalty* = 0.0)

De Jong Function: Unimodal, n-dimensional.

Optimal example:

u = [0, 0, 0, 0, ... n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.

penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.5 def ObjectiveFunction.ObjectiveFunction.easom (*self, u, penalty =* `0.0`)

Easom Function: Multimodal, n-dimensional.

Optimal example:

u = [pi, pi]

fitness = 1.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.6 def ObjectiveFunction.ObjectiveFunction.griewank (*self, u, penalty =* `0.0`)

Griewank Function: Multimodal, n-dimensional.

Optimal example:

u = [0, 0, 0, ..., 0]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.

u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.7 def ObjectiveFunction.ObjectiveFunction.mi_chemical_process (*self, u, penalty =* 1E15)

Chemical process design mixed integer problem.

Optimal example:

u = [(0.2, 0.8, 1.907878, 1, 1, 0, 1]

fitness = 4.579582

Taken from: "An Improved PSO Algorithm for Solving Non-convex NLP/MINLP Problems with Equality Constraints"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated. [x1, x2, x3, y1, y2, y3, y4]
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.8 def ObjectiveFunction.ObjectiveFunction.mi_pressure_vessel (*self, u, penalty =* 1E15)

Mixed Integer Pressure vessel objective function with penalty method of constraint enforcement.

Near optimal example:

u = [58.2298, 44.0291, 17, 9]

fitness = 7203.24

Optimal example obtained with Gnowee:

u = [38.819876, 221.985576, 0.750000, 0.375000]

fitness = 5855.893191

Taken from: "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.9 def ObjectiveFunction.ObjectiveFunction.mi_spring (*self*, *u*, *penalty* = `1E15`)

Spring objective function with penalty method of constraint enforcement.

Optimal Example:

u = [1.22304104, 9, 36] = [1.22304104, 9, 0.307]

fitness = 2.65856

Taken from Lampinen, "Mixed Integer-Discrete-Continuous Optimization by Differential Evolution"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.10 def ObjectiveFunction.ObjectiveFunction.pressure_vessel (*self*, *u*, *penalty* = `1E15`)

Pressure vessel objective function with penalty method of constraint enforcement.

Near Optimal Example:

u = [0.81250000001, 0.4375, 42.098445595854923, 176.6365958424394]

fitness = 6059.714335

Optimal obtained using Gnowee:

u = [0.7781686880924992, 0.3846491857203429, 40.319621144688995, 199.99996630362293]

fitness = 5885.33285347

Taken from: "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.11 def ObjectiveFunction.ObjectiveFunction.rastrigin (*self*, *u*, *penalty* = 0.0)

Rastrigin Function: Multimodal, n-dimensional.

Optimal example:

u = [0, 0, 0, ..., 0]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.12 def ObjectiveFunction.ObjectiveFunction.rosenbrock (*self*, *u*, *penalty* = 0.0)

Rosenbrock Function: uni-modal, n-dimensional.

Optimal example:

u = [1, 1, 1, ..., 1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.13 def ObjectiveFunction.ObjectiveFunction.set_obj_func (*self, funcName*)

Converts an input string name for a function to a function handle.

Parameters

self	*pointer* The ObjectiveFunction pointer.
funcName	*string* A string identifying the objective function to be used.

9.4.3.14 def ObjectiveFunction.ObjectiveFunction.shifted_ackley (*self, u, penalty* = `0.0`)

Ackley Function: Mulitmodal, n dimensional Ackley Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.

Optimal example:

u = [0, 1, 2, 3, ... n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.15 def ObjectiveFunction.ObjectiveFunction.shifted_dejong (*self*, *u*, *penalty* = 0.0)

De Jong Function: Unimodal, n-dimensional De Jong Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.

Optimal example:

u = [0, 1, 2, 3, ... n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Taken from: "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.16 def ObjectiveFunction.ObjectiveFunction.shifted_easom (*self*, *u*, *penalty* = 0.0)

Easom Function: Multimodal, n-dimensional Easom Function that is shifted from the symmetric pi, pi optimimum.

Optimal example:

u = [pi, pi+1]

fitness = 1.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.

penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.17 def ObjectiveFunction.ObjectiveFunction.shifted_griewank (*self, u, penalty =* 0.0)

Griewank Function: Multimodal, n-dimensional Griewank Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.

Optimal example:

u = [0, 1, 2, ..., n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.18 def ObjectiveFunction.ObjectiveFunction.shifted_rastrigin (*self, u, penalty =* 0.0)

Rastrigin Function: Multimodal, n-dimensional Rastrigin Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimimum.

Optimal example:

u = [0, 1, 2, ..., n-1]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.19 def ObjectiveFunction.ObjectiveFunction.shifted_rosenbrock (*self, u, penalty =* `0.0`)

Rosenbrock Function: uni-modal, n-dimensional Rosenbrock Function that is shifted from the symmetric 0,0,0...0 optimimum.

Optimal example:

u = [1, 2, 3, ...n]

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.20 def ObjectiveFunction.ObjectiveFunction.speed_reducer (*self, u, penalty =* `1E15`)

Speed reducer objective function with penalty method of constraint enforcement.

Optimal example:

u = [58.2298, 44.0291, 17, 9]

fitness = 2996.34784914

Optimal example obtained with Gnowee:

u = [3.500000, 0.7, 17, 7.300000, 7.800000, 3.350214, 5.286683]

fitness = 5855.893191

Taken from: "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.21 def ObjectiveFunction.ObjectiveFunction.spring (*self*, *u*, *penalty* = `1E15`)

Spring objective function with penalty method of constraint enforcement.

Optimal Example:

u = [0.05169046, 0.356750, 11.287126]

fitness = 0.0126653101469

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.22 def ObjectiveFunction.ObjectiveFunction.tsp (*self*, *u*, *penalty* = `0.0`)

Generic objective funtion to evaluate the TSP optimization by calculating total distance traveled.

Parameters

self	*pointer* The ObjectiveFunction pointer.

u	*array* The city pairs to be evaluated.
penalty	*float* Per constraint violation penalty. Not used for this function; included for method input consistency.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.3.23 def ObjectiveFunction.ObjectiveFunction.welded_beam (*self, u, penalty =* `1E15`)

Welded Beam objective function with penalty method of constraint enforcement.

Optimal Example:

u = [0.20572965, 3.47048857, 9.0366249, 0.20572965]

fitness = 1.7248525603892848

Taken from: "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer"

Parameters

self	*pointer* The ObjectiveFunction pointer.
u	*array* The design parameters to be evaluated.
penalty	*float* Per constraint violation penalty.

Returns

array: The fitness associated with the specified input.
array: The assessed value for each constraint for the specified input.

9.4.4 Member Data Documentation

9.4.4.1 ObjectiveFunction.ObjectiveFunction.func

function handle The function handle for the objective function to be used for the optimization.

The function must be specified as a method of the class.

9.4.4.2 ObjectiveFunction.ObjectiveFunction.objective

integer, float, or numpy array The desired outcome of the optimization.

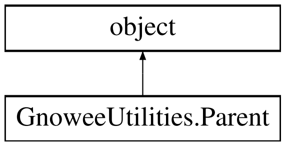
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ObjectiveFunction.py

9.5 GnoweeUtilities.Parent Class Reference

The class contains all of the parameters pertinent to a member of the population.

Inheritance diagram for GnoweeUtilities.Parent:



Public Member Functions

- def __init__

  Constructor to build the Parent class.
- def __repr__

  Parent print function.
- def __str__

  Human readable Parent print function.

Public Attributes

- variables

  array: The set of variables representing a design solution.
- fitness

  float: The assessed fitness for the current set of variables.
- changeCount

  integer: The number of improvements to the current population member.
- stallCount

  integer: he number of evaluations since the last improvement.

9.5.1 Detailed Description

The class contains all of the parameters pertinent to a member of the population.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 def GnoweeUtilities.Parent.__init__ (*self*, *variables* = `None`, *fitness* = `1E15`, *changeCount* = `0`, *stallCount* = `0`)

Constructor to build the Parent class.

Parameters

self	*Parent pointer* The Parent pointer.

variables	*array* The set of variables representing a design solution.
fitness	*float* The assessed fitness for the current set of variables.
changeCount	*integer* The number of improvements to the current population member.
stallCount	*integer* The number of evaluations since the last improvement.

9.5.3 Member Function Documentation

9.5.3.1 def GnoweeUtilities.Parent.__repr__ (*self*)

Parent print function.

Parameters

self	*Parent pointer* The Parent pointer.

9.5.3.2 def GnoweeUtilities.Parent.__str__ (*self*)

Human readable Parent print function.

Parameters

self	*Parent pointer* The Parent pointer.

9.5.4 Member Data Documentation

9.5.4.1 GnoweeUtilities.Parent.changeCount

integer: The number of improvements to the current population member.

9.5.4.2 GnoweeUtilities.Parent.fitness

float: The assessed fitness for the current set of variables.

9.5.4.3 GnoweeUtilities.Parent.stallCount

integer: he number of evaluations since the last improvement.

9.5.4.4 GnoweeUtilities.Parent.variables

array: The set of variables representing a design solution.

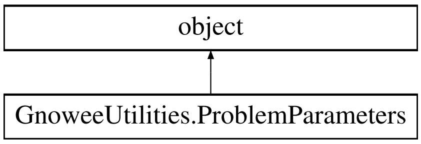
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py

9.6 GnoweeUtilities.ProblemParameters Class Reference

Creates an object containing key features of the chosen optimization problem.

Inheritance diagram for GnoweeUtilities.ProblemParameters:



Public Member Functions

- def __init__
- def __repr__

    ProblemParameters class attribute print function.
- def __str__

    Human readable ProblemParameters print function.
- def get_preset_params

    Instantiates a ProblemParameters object and populations member variables from a set of predefined problem types.

Public Attributes

- lb

    array: The lower bounds of the design variable(s).
- ub

    array: The upper bounds of the design variable(s).
- varType

    array: The type of variable for each position in the upper and lower bounds array.
- discreteVals

    array: nxm with n=# of discrete variables and m=# of values that can be taken for each variable.
- optimum

    float: The global optimal solution.
- pltTitle

    string: The title used for plotting the results of the optimization.
- histTitle

    string: The plot title for the histogram of the optimization results.
- varNames

    list of strings: The names of the variables for the optimization problem.

9.6.1 Detailed Description

Creates an object containing key features of the chosen optimization problem.

The methods provide a way of predefining problems for repeated use.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 def GnoweeUtilities.ProblemParameters.__init__ (*self*, *lowerBounds* = [], *upperBounds* = [], *varType* = [], *discreteVals* = [], *optimum* = 0.0, *pltTitle* = '', *histTitle* = '', *varNames* = '')

Parameters

self	*pointer* The ProblemParameters pointer.
lowerBounds	*array* The lower bounds of the design variable(s). Only enter the bounds for continuous and integer/binary variables. The order must match the order specified in varType and ub.
upperBounds	*array* The upper bounds of the design variable(s). Only enter the bounds for continuous and integer/binary variables. The order must match the order specified in varType and lb.
varType	*list or array* The type of variable for each position in the upper and lower bounds array. Discrete variables are to be included last as they are specified separatly from the lb/ub throught the discreteVals optional input. A variable can have two types (for example, 'dx' could denote a layer that can take multiple materials and be placed at multiple design locations) Allowed values: 'c' = continuous over a given range (range specified in lb & ub). 'i' = integer/binary (difference denoted by ub/lb). 'd' = discrete where the allowed values are given by the option discreteVals nxm arrary with n=# of discrete variables and m=# of values that can be taken for each variable. 'x' = combinatorial. All of the variables denoted by x are assumed to be "swappable" in combinatorial permutations. There must be at least two variables denoted as combinatorial. 'f' = fixed design variable. Will not be considered of any permutation.
discreteVals	*list of list(s)* nxm with n=# of discrete variables and m=# of values that can be taken for each variable. For example, if you had two variables representing the tickness and diameter of a cylinder that take standard values, the discreteVals could be specified as: discreteVals = [[0.125, 0.25, 0.375], [0.25, 0.5, 075]] Gnowee will then map the optimization results to these allowed values.
optimum	*float* The global optimal solution.
pltTitle	*string* The title used for plotting the results of the optimization.
histTitle	*string* The plot title for the histogram of the optimization results.
varNames	*list of strings* The names of the variables for the optimization problem.

9.6.3 Member Function Documentation

9.6.3.1 def GnoweeUtilities.ProblemParameters.__repr__ (*self*)

ProblemParameters class attribute print function.

Parameters

self	*pointer* The ProblemParameters pointer.

9.6.3.2 def GnoweeUtilities.ProblemParameters.__str__ (*self*)

Human readable ProblemParameters print function.

Parameters

self	*pointer* The ProblemParameters pointer.

9.6.3.3 def GnoweeUtilities.ProblemParameters.get_preset_params (*self, funct, algorithm = '', dimension =* 2)

Instantiates a ProblemParameters object and populations member variables from a set of predefined problem types.

Parameters

self	*pointer* The ProblemParameters pointer.
funct	*string* Name of function being optimized.
algorithm	*string* Name of optimization program used.
dimension	*integer* Used to set the dimension for scalable problems.

Returns

ProblemParameters object: A ProblemParameters object with populated member variables.

9.6.4 Member Data Documentation

9.6.4.1 GnoweeUtilities.ProblemParameters.discreteVals

array: nxm with n=# of discrete variables and m=# of values that can be taken for each variable.

9.6.4.2 GnoweeUtilities.ProblemParameters.histTitle

string: The plot title for the histogram of the optimization results.

9.6.4.3 GnoweeUtilities.ProblemParameters.lb

array: The lower bounds of the design variable(s).

9.6.4.4 GnoweeUtilities.ProblemParameters.optimum

float: The global optimal solution.

9.6.4.5 GnoweeUtilities.ProblemParameters.pltTitle

string: The title used for plotting the results of the optimization.

9.6.4.6 def GnoweeUtilities.ProblemParameters.ub

array: The upper bounds of the design variable(s).

9.6.4.7 GnoweeUtilities.ProblemParameters.varNames

list of strings: The names of the variables for the optimization problem.

9.6.4.8 GnoweeUtilities.ProblemParameters.varType

array: The type of variable for each position in the upper and lower bounds array.

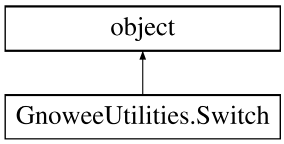
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py

9.7 GnoweeUtilities.Switch Class Reference

Creates a switch class object to switch between cases.

Inheritance diagram for GnoweeUtilities.Switch:



Public Member Functions

- def __init__

  Case constructor.
- def __iter__

  Return the match method once, then stop.
- def match

  Indicate whether or not to enter a case suite.

Public Attributes

- value

  string: Case selector value.
- fall

  boolean: Match indicator.

9.7.1 Detailed Description

Creates a switch class object to switch between cases.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 def GnoweeUtilities.Switch.__init__ (*self, value*)

Case constructor.

Parameters

self	*pointer* The Switch pointer.
value	*string* Case selector value.

9.7.3 Member Function Documentation

9.7.3.1 def GnoweeUtilities.Switch.__iter__ (*self*)

Return the match method once, then stop.

Parameters

self	*pointer* The Switch pointer.

9.7.3.2 def GnoweeUtilities.Switch.match (*self, args*)

Indicate whether or not to enter a case suite.

Parameters

self	*pointer* The Switch pointer.
∗args	*list* List of comparisons.

Returns

boolean: Outcome of comparison match

9.7.4 Member Data Documentation

9.7.4.1 GnoweeUtilities.Switch.fall

boolean: Match indicator.

9.7.4.2 GnoweeUtilities.Switch.value

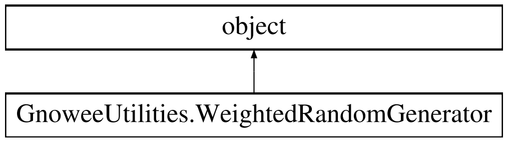
string: Case selector value.

The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py

9.8 GnoweeUtilities.WeightedRandomGenerator Class Reference

Inheritance diagram for GnoweeUtilities.WeightedRandomGenerator:



Public Member Functions

- def **__init__**
- def **next**
- def **__call__**

Public Attributes

- **totals**

9.8.1 Detailed Description

```
Defines a class of weights to be used to select number of instances in array randomly with
linear weighting.

Parameters
==========
self : object
    Current instance of the class
weights : array
    The array of weights (Higher = more likely to be selected)

Returns
=======
bisect.bisect_right(self.totals, rnd) : integer
    The randomly selected index of the weights array
```

The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py

Chapter 10

File Documentation

10.1 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Constraints.py File Reference

Classes

- class Constraints.Constraint

  The class creates a Constraints object that can be used in optimization algorithms.

10.2 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Gnowee.py File Reference

Namespaces

- Gnowee

  Contains the Gnowee optimization program and associated utilities.

Functions

- def Gnowee.main

  Main controller program for the Gnowee optimization.

10.3 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeHeuristics.py File Reference

Classes

- class GnoweeHeuristics.GnoweeHeuristics

  The class is the foundation of the Gnowee optimization algorithm.

Namespaces

- Gnowee

  Contains the Gnowee optimization program and associated utilities.

Functions

- def GnoweeHeuristics.simple_bounds

    Application of problem boundaries to generated solutions.
- def GnoweeHeuristics.rejection_bounds

    Application of problem boundaries to generated solutions.

10.4 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py File Reference

Classes

- class GnoweeUtilities.Parent

    The class contains all of the parameters pertinent to a member of the population.
- class GnoweeUtilities.Event

    Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback.
- class GnoweeUtilities.ProblemParameters

    Creates an object containing key features of the chosen optimization problem.
- class GnoweeUtilities.Switch

    Creates a switch class object to switch between cases.
- class GnoweeUtilities.WeightedRandomGenerator

Namespaces

- Gnowee

    Contains the Gnowee optimization program and associated utilities.

Functions

- def **GnoweeUtilities.Get_Best**

10.5 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Objective-Function.py File Reference

Classes

- class ObjectiveFunction.ObjectiveFunction

    The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Namespaces

- Gnowee

    Contains the Gnowee optimization program and associated utilities.

Functions

- def ObjectiveFunction.prod

    Computes the product of a set of numbers (ie big PI, mulitplicative equivalent to sum).

10.6 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/OptiPlot.py File Reference

Namespaces

- Gnowee

  Contains the Gnowee optimization program and associated utilities.

Functions

- def OptiPlot.plot_vars

  Plot the variables as they change in the optimization process.
- def OptiPlot.plot_hist

  Plots the histogram of function evaluation results from multiple runs of an optimization algorithm.
- def OptiPlot.plot_hist_comp

  Histograms and plots the comparison of two sets of function evaluation data.
- def OptiPlot.plot_feval_hist

  Plots the fitness vs function evaluation results of an optimization algorithm run.
- def OptiPlot.plot_tlf

  Plots a comparison of the TLF to the Levy distribution.
- def OptiPlot.plot_optimization

  Plots the results of optimization process for a given algorithm and parameter.

10.7 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Sampling.py File Reference

Namespaces

- Gnowee

  Contains the Gnowee optimization program and associated utilities.

Functions

- def Sampling.initial_samples

  Generate a set of samples in a given phase space.
- def Sampling.plot_samples

  Plot the first 2 and 3 dimensions on the sample distribution.
- def Sampling.levy

  Sample the Levy distribution given by.
- def Sampling.tlf

  Samples from a truncated Levy flight distribution (TLF) according to Manegna, "Stochastic Process with Ultraslow Convergence to a Gaussian: The Truncated Levy Flight" to map a levy distribution onto the interval [0,1].
- def Sampling.NOLH

  This library allows to generate Nearly Orthogonal Latin Hypercubes (NOLH) according to Cioppa (2007) and De Rainville et al.
- def Sampling.params

  Returns the NOLH order m, the required configuration length q and the number of columns to remove to obtain the desired dimensionality.
- def Sampling.get_cdr_permutations

  Generate a set of CDR permulations for NOLH.

Index