Gnowee

Generated by Doxygen 1.8.6

Sun May 14 2017 12:54:04

Contents

1	Mair	Page Control of the C	1
2	Mod	ule Index	3
	2.1	Modules	3
3	Nam	espace Index	5
	3.1	Packages	5
4	Hier	archical Index	7
	4.1	Class Hierarchy	7
5	Clas	s Index	9
	5.1	Class List	9
6	File	ndex	11
	6.1	File List	11
7	Mod	ule Documentation	13
	7.1	Constraints	13
		7.1.1 Detailed Description	13
	7.2	Gnowee	14
		7.2.1 Detailed Description	14
		7.2.2 Function Documentation	14
		7.2.2.1 main	14
	7.3	GnoweeHeuristics	15
		7.3.1 Detailed Description	15
		7.3.2 Function Documentation	15
		7.3.2.1 rejection_bounds	15
		7.3.2.2 simple_bounds	16
	7.4	GnoweeUtilities	17
		7.4.1 Detailed Description	17
		7.4.2 Function Documentation	17
		7.4.2.1 init	17
	7.5	Objective Function	18

iv CONTENTS

		7.5.1	Detailed Description	18
		7.5.2	Function Documentation	18
			7.5.2.1 prod	18
	7.6	OptiPlo	t	19
		7.6.1	Detailed Description	19
		7.6.2	Function Documentation	19
			7.6.2.1 plot_feval_hist	19
			7.6.2.2 plot_hist	20
			7.6.2.3 plot_hist_comp	20
			7.6.2.4 plot_optimization	20
			7.6.2.5 plot_tlf	21
			7.6.2.6 plot_vars	21
	7.7	Sampli	ng 2	22
		7.7.1	Detailed Description	22
		7.7.2	Function Documentation	22
			7.7.2.1 get_cdr_permutations	22
			7.7.2.2 initial_samples	23
			7.7.2.3 levy	23
			7.7.2.4 NOLH	24
			7.7.2.5 params	24
			7.7.2.6 plot_samples	25
			7.7.2.7 tlf	25
	7.8	TSP .		26
		7.8.1	Detailed Description	26
8	Nam	oenaco	Documentation 2	27
Ů	8.1			27
	0.1	8.1.1		- 1 27
		0.1.1	Detailed Description	-1
9	Clas	s Docui	nentation	29
	9.1	Constr	ints.Constraint Class Reference	29
		9.1.1	Detailed Description	30
		9.1.2	Constructor & Destructor Documentation	30
			9.1.2.1init	30
		9.1.3	Member Function Documentation	30
			9.1.3.1repr	30
			9.1.3.2str	31
			9.1.3.3 get_penalty	31
			9.1.3.4 greater_than	31
			9.1.3.5 less_or_equal	31
			9.1.3.6 less_than	32

CONTENTS

		9.1.3.7	mi_chemical_process	32
		9.1.3.8	mi_pressure_vessel	32
		9.1.3.9	mi_spring	33
		9.1.3.10	pressure_vessel	33
		9.1.3.11	set_constraint_func	34
		9.1.3.12	speed_reducer	34
		9.1.3.13	spring	35
		9.1.3.14	welded_beam	35
	9.1.4	Member	Data Documentation	36
		9.1.4.1	constraint	36
		9.1.4.2	func	36
9.2	Gnowe	eUtilities.E	Event Class Reference	36
	9.2.1	Detailed	Description	36
	9.2.2	Construc	tor & Destructor Documentation	37
		9.2.2.1	init	37
	9.2.3	Member	Function Documentation	37
		9.2.3.1	repr	37
		9.2.3.2	str	37
	9.2.4	Member	Data Documentation	37
		9.2.4.1	design	37
		9.2.4.2	evaluations	37
		9.2.4.3	fitness	38
		9.2.4.4	generation	38
9.3	Gnowe	eHeuristic	s.GnoweeHeuristics Class Reference	38
	9.3.1	Detailed	Description	39
	9.3.2	Construc	tor & Destructor Documentation	39
		9.3.2.1	init	39
	9.3.3	Member	Function Documentation	41
		9.3.3.1	repr	41
		9.3.3.2	<u>str</u>	41
		9.3.3.3	cont_levy_flight	41
		9.3.3.4	crossover	41
		9.3.3.5	disc_levy_flight	43
		9.3.3.6	elite_crossover	43
		9.3.3.7	initialize	44
		9.3.3.8	mutate	45
		9.3.3.9	population_update	45
		9.3.3.10	scatter_search	46
	9.3.4	Member	Data Documentation	47
		9.3.4.1	alpha	47

vi CONTENTS

		9.3.4.2	convTol	47
		9.3.4.3	fracDiscovered	47
		9.3.4.4	fracElite	47
		9.3.4.5	fracLevy	47
		9.3.4.6	gamma	47
		9.3.4.7	initSampling	47
		9.3.4.8	maxFevals	47
		9.3.4.9	maxGens	47
		9.3.4.10	$n \ldots \ldots \ldots \ldots \ldots$	48
		9.3.4.11	optConvTol	48
		9.3.4.12	penalty	48
		9.3.4.13	population	48
		9.3.4.14	scalingFactor	48
		9.3.4.15	stallLimit	48
9.4	Objecti	iveFunction	n.ObjectiveFunction Class Reference	48
	9.4.1	Detailed	Description	50
	9.4.2	Construc	tor & Destructor Documentation	50
		9.4.2.1	init	50
	9.4.3	Member	Function Documentation	50
		9.4.3.1	repr	50
		9.4.3.2	str	50
		9.4.3.3	ackley	50
		9.4.3.4	dejong	52
		9.4.3.5	easom	52
		9.4.3.6	griewank	53
		9.4.3.7	mi_chemical_process	53
		9.4.3.8	mi_pressure_vessel	54
		9.4.3.9	mi_spring	54
		9.4.3.10	pressure_vessel	54
		9.4.3.11	rastrigin	55
		9.4.3.12	rosenbrock	55
		9.4.3.13	set_obj_func	56
		9.4.3.14	shifted_ackley	56
		9.4.3.15	shifted_dejong	56
		9.4.3.16	shifted_easom	57
		9.4.3.17	shifted_griewank	57
		9.4.3.18	shifted_rastrigin	58
		9.4.3.19	shifted_rosenbrock	58
		9.4.3.20	speed_reducer	59
		9.4.3.21	spring	59

CONTENTS vii

		9.4.3.22	tsp	60
		9.4.3.23	welded_beam	60
	9.4.4	Member E	Data Documentation	60
		9.4.4.1	func	60
		9.4.4.2	objective	60
9.5	Gnowe	eUtilities.Pa	arent Class Reference	61
	9.5.1	Detailed D	Description	61
	9.5.2	Construct	or & Destructor Documentation	61
		9.5.2.1	init	61
	9.5.3	Member F	Function Documentation	62
		9.5.3.1	repr	62
		9.5.3.2	str	62
	9.5.4	Member D	Data Documentation	62
		9.5.4.1	changeCount	62
		9.5.4.2	fitness	62
		9.5.4.3	stallCount	62
		9.5.4.4	variables	62
9.6	Gnowe	eUtilities.P	roblemParameters Class Reference	63
	9.6.1	Detailed D	Description	64
	9.6.2	Construct	or & Destructor Documentation	64
		9.6.2.1	init	64
	9.6.3	Member F	Function Documentation	65
		9.6.3.1	repr	65
		9.6.3.2	str	65
		9.6.3.3	map_from_discretes	66
		9.6.3.4	map_to_discretes	66
		9.6.3.5	sanitize_inputs	66
		9.6.3.6	set_preset_params	66
	9.6.4	Member E	Data Documentation	67
		9.6.4.1	cID	67
		9.6.4.2	constraints	67
		9.6.4.3	dID	67
		9.6.4.4	discreteVals	67
		9.6.4.5	histTitle	67
		9.6.4.6	iID	67
		9.6.4.7	$lb \ \ldots \ $	67
		9.6.4.8	objective	68
		9.6.4.9	optimum	68
		9.6.4.10	pltTitle	68
		9.6.4.11	ub	68

viii CONTENTS

			9.6.4.12 varNames	68
			9.6.4.13 varType	68
			9.6.4.14 xID	68
	9.7	Gnowe	eUtilities.Switch Class Reference	68
		9.7.1	Detailed Description	69
		9.7.2	Member Function Documentation	69
			9.7.2.1iter	69
			9.7.2.2 match	69
		9.7.3	Member Data Documentation	69
			9.7.3.1 fall	69
			9.7.3.2 value	70
	9.8	TSP.TS	P Class Reference	70
		9.8.1	Detailed Description	70
		9.8.2	Constructor & Destructor Documentation	70
			9.8.2.1init	70
		9.8.3	Member Function Documentation	71
			9.8.3.1repr	71
			9.8.3.2str	71
			9.8.3.3 build_prob_params	71
			9.8.3.4 read_tsp	71
		9.8.4	Member Data Documentation	72
			9.8.4.1 dimension	72
			9.8.4.2 name	72
			9.8.4.3 nodes	72
			9.8.4.4 optimum	72
	9.9	Sampli	ng.WeightedRandomGenerator Class Reference	72
		9.9.1	Detailed Description	73
		9.9.2	Constructor & Destructor Documentation	73
			9.9.2.1init	73
		9.9.3	Member Function Documentation	73
			9.9.3.1call	73
			9.9.3.2 next	73
		9.9.4	Member Data Documentation	74
			9.9.4.1 totals	74
10	Eile !	Docum-	entation	75
10				
				75 75
				75
	10.3		pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeHeuristics.py File Ref-	75

CONTENTS

Ind	lex		79
	10.8	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/TSP.py File Reference	78
	10.7	$/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Sampling.py\ File\ Reference .$	77
	10.6	$/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/OptiPlot.py\ File\ Reference \ .\ .$	77
	10.5	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ObjectiveFunction.py File Reference	76
	10.4	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py File Reference	76

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

2 Main Page

james.e.bevins@gmail.com

Licensing Information

Acknowledements

AF, advisor, NSF

Module Index

2.1 Modules

Here is a list of all modules:

Constraints									 						 								13
Gnowee																							14
GnoweeHeuristic	cs																						15
GnoweeUtilities									 														17
ObjectiveFunctio	n								 														18
OptiPlot																							19
Sampling																							22
TSP									 						 								26

Module Index

Namespace Index

3.1	Packag	nes
J. I	i achai	463

Here are the packages with brief descriptions (if available):	
Gnowee	
Contains the Growee ontimization program and associated utilities	27

6 Namespace Index

Hierarchical Index

4.1 Class Hierarchy

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

bject	
Constraints.Constraint	 29
GnoweeUtilities.Event	 36
GnoweeUtilities.Parent	 61
GnoweeUtilities.ProblemParameters	 63
GnoweeHeuristics.GnoweeHeuristics	 38
GnoweeUtilities.Switch	 68
ObjectiveFunction.ObjectiveFunction	 48
Sampling.WeightedRandomGenerator	 72
SP.TSP	 70

8 **Hierarchical Index**

Class Index

5.1 Class List

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

Constraints.Constraint Constraint	
The class creates a Constraints object that can be used in optimization algorithms	29
GnoweeUtilities.Event	
Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback	36
GnoweeHeuristics.GnoweeHeuristics	
The class is the foundation of the Gnowee optimization algorithm	38
ObjectiveFunction. ObjectiveFunction	
This class creates a ObjectiveFunction object that can be used in optimization algorithms	48
GnoweeUtilities.Parent	
The class contains all of the parameters pertinent to a member of the population	61
GnoweeUtilities.ProblemParameters	
Creates an object containing key features of the chosen optimization problem	63
GnoweeUtilities.Switch	
Creates a switch class object to switch between cases	68
TSP.TSP	
This class creates a TSP object that can be used in optimization algorithms to solve the Travelling	
Saleman Problem	70
Sampling.WeightedRandomGenerator	
Defines a class of weights to be used to select based on linear weighting	72

10 Class Index

File Index

6.1 File List

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

/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Constraints.py	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Gnowee.py	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeHeuristics.py	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py	76
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ObjectiveFunction.py	76
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/OptiPlot.py	77
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Sampling.py	77
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/TSP.py	78

12 File Index

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

12May17

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

13May17

7.2.2 Function Documentation

7.2.2.1 def Gnowee.main (gh)

Main controller program for the **Gnowee** optimization.

Parameters

gh	GnoweeHeuristic object
	An object constaining the problem definition and the settings and methods required for the
	Gnowee optimization algorithm.

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 15

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

14May17

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.

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 17

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.

Functions

def GnoweeUtilities.Switch.__init__

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

1May17

7.4.2 Function Documentation

7.4.2.1 def GnoweeUtilities.Switch.__init__ (self, value)

Creates a switch class object to switch between cases.

Case constructor.

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

7.5 ObjectiveFunction

Defines a class to perform objective function calculations.

Classes

· class ObjectiveFunction.ObjectiveFunction

This 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

12May17

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

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.

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.

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

7.6 OptiPlot 21

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.

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.

Classes

· class Sampling.WeightedRandomGenerator

Defines a class of weights to be used to select based on linear weighting.

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

12May17

7.7.2 Function Documentation

7.7.2.1 def Sampling.get_cdr_permutations (dim)

Generate a set of CDR permulations for NOLH.

7.7 Sampling 23

Parameters

dim	integer
	The dimension of the space.

Returns

array: A configuration vector.

array: Array containing the indexes of the columnns 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."

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.

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 q-1] or [1 q] 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 columnns 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.

dim	integer
	The dimension of the space.

7.7 Sampling 25

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).

7.8 TSP

Defines a class to perform Travelling Salesman Problem (TSP) optimization.

Classes

• class TSP.TSP

This class creates a TSP object that can be used in optimization algorithms to solve the Travelling Saleman Problem.

7.8.1 Detailed Description

Defines a class to perform Travelling Salesman Problem (TSP) optimization. This class is designed to initialize and store TSP problems from the TSPLIB database. It will read in standard TSPLIB files, and create a TSP object for use in optimization routines.

Author

James Bevins

Date

13May17

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

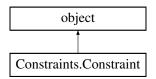


Chapter 9

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 spring

Spring penalty method of constraint enforcement.

· def mi_spring

Spring penalty method of constraint enforcement.

· def welded_beam

Welded Beam penalty method of constraint enforcement.

· def pressure vessel

Pressure vessel penalty method of constraint enforcement.

def mi_pressure_vessel

Mixed Integer Pressure vessel penalty method of constraint enforcement.

· def speed reducer

Speed reducer penalty method of constraint enforcement.

def mi_chemical_process

Chemical process design constraint enforcement.

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.

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.mi_chemical_process (self, u)

Chemical process design constraint enforcement.

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.
и	array The design parameters to be evaluated. [x1, x2, x3, y1, y2, y3, y4]

Returns

array: The fitness associated with the specified input.

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

9.1.3.8 def Constraints.Constraint.mi_pressure_vessel (self, u)

Mixed Integer Pressure vessel 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.

Returns

array: The fitness associated with the specified input.

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

9.1.3.9 def Constraints.Constraint.mi_spring (self, u)

Spring 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.
и	array
	The design parameters to be evaluated.

Returns

float: The assessed penalty for constraint violations for the specified input.

9.1.3.10 def Constraints.Constraint.pressure_vessel (self, u)

Pressure vessel 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.
и	array
	The design parameters to be evaluated.

Returns

array: The fitness associated with the specified input.

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

9.1.3.11 def Constraints.Constraint.set_constraint_func (self, funcName)

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

Parameters

pointer
The Constraint pointer.
string
A string identifying the constraint function to be used.
3 , 3

9.1.3.12 def Constraints.Constraint.speed_reducer (self, u)

Speed reducer 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"

self	pointer
	The ObjectiveFunction pointer.

и	array
	The design parameters to be evaluated.

Returns

array: The fitness associated with the specified input.

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

9.1.3.13 def Constraints.Constraint.spring (self, u)

Spring penalty method of constraint enforcement.

Optimal Example:

u = [0.05169046, 0.356750, 11.287126]

fitness = 0.0126653101469

Parameters

self	pointer
	The ObjectiveFunction pointer.
и	array
	The design parameters to be evaluated.

Returns

array: The fitness associated with the specified input.

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

9.1.3.14 def Constraints.Constraint.welded_beam (self, u)

Welded Beam 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.
и	array The design parameters to be evaluated.

Returns

array: The fitness associated with the specified input.

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

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.

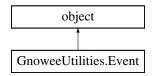
The function must be specified as a method of the class.

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.

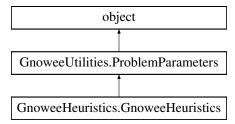
The documentation for this class was generated from the following file:

/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.

def population_update

Calculate fitness, apply constraints, if present, and update the population if the children are better than their parents.

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.

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, optConvTol = 1e-2, kwargs )
```

Constructor to build the GnoweeHeuristics class.

This class must be fully instantiated to run the Gnowee program. It consists of 2 main parts: The main class attributes and the inhereted ProblemParams class attributes. The main class attributes contain defaults that don't require direct user input to work (but can be modified by user input if desired), but the ProblemParameters class does require proper instantiation by the user.

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>="">.

2016	Consumal la unistic mainten
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
"doEinto	Elite fraction probability used for the scatter_search(), crossover(), and cont_crossover()
	heuristics.
	Heurisics.
frool over	float
fracLevy	
	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.
	The man so to take the same to seposite tenesion.
scalingFactor	float
Scaling actor	Step size scaling factor used to adjust Levy flights to length scale of system. The implemen-
	tation 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.
	I

optConvTol	float
	The maximum deviation from the best know fitness value before the search terminates.
kwargs	ProblemParameters class arguments Keyword arguments for the attributes of the ProblemParameters class. If not provided. The inhereted attributes will be set to the class defaults.

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)

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.
рор	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. *list:* A list of the identities of the chosen index for each child.

9.3.3.4 def GnoweeHeuristics.GnoweeHeuristics.crossover (self, pop)

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.
	· ·
рор	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. *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)

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.
рор	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. *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.
рор	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)

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().

Returns

list of arrays: The initialized set of samples.

9.3.3.8 def GnoweeHeuristics.GnoweeHeuristics.mutate (self, pop)

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.
рор	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.

9.3.3.9 def GnoweeHeuristics.GnoweeHeuristics.population_update (self, parents, children, timeline = None, genUpdate = 0, adoptedParents = [], mhFrac = 0.0, randomParents = False)

Calculate fitness, apply constraints, if present, and update the population if the children are better than their parents. Several optional inputs are available to modify this process. Refer to the input param documentation for more details.

parents	list of parent objects
	The current parents representing system designs.
children	list of arrays
	The children design variables representing new system designs.

timeline	list of history objects The histories of the optimization process containing best design, fitness, generation, and function evaluations.
genUpdate	integer Indicator for how many generations to increment the counter by. Genenerally 0 or 1.
adoptedParents	list A list of alternative parents to compare the children against. This alternative parents are then held accountable for not being better than the children of others.
mhFrac	float The Metropolis-Hastings fraction. A fraction of the otherwise discarded parents will be evaluated for acceptance against the greater population.
randomParents	boolean If True, a random parent will be selected for comparison to the children. No one is safe.

Returns

list of parent objects: The current parents representing system designs.

integer: The number of replacements made.

list of history objects: If an initial timeline was provided, returns an updated history of the optimization process containing best design, fitness, generation, and function evaluations.

9.3.3.10 def GnoweeHeuristics.GnoweeHeuristics.scatter_search (self, pop)

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.

self	GnoweeHeuristic pointer
	The GnoweeHeuristics pointer.

рор	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. *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.penalty

float: Individual constraint violation penalty to add to objective function.

9.3.4.13 GnoweeHeuristics.GnoweeHeuristics.population

integer: The number of members in each generation.

9.3.4.14 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.

9.3.4.15 GnoweeHeuristics.GnoweeHeuristics.stallLimit

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

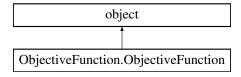
The documentation for this class was generated from the following file:

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

9.4 ObjectiveFunction.ObjectiveFunction Class Reference

This 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.

def mi_spring

Spring objective function.

· def welded_beam

Welded Beam objective function.

· def pressure vessel

Pressure vessel objective function.

• def mi_pressure_vessel

Mixed Integer Pressure vessel objective function.

· def speed_reducer

Speed reducer objective function.

• 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

 $\label{thm:continuous} \textit{Generic objective funtion to evaluate the TSP optimization by calculating total distance traveled.}$

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

This 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.

This class specifies the objective function to be used for a optimization process.

Parameters

self	ObjectiveFunction 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 optimization function chosen. This is used in objective functions that involve a comparison against a desired outcome.

9.4.3 Member Function Documentation

9.4.3.1 def ObjectiveFunction.ObjectiveFunction.__repr__ (self)

ObjectiveFunction class param print function.

Parameters

self	ObjectiveFunction pointer
	The ObjectiveFunction pointer.

9.4.3.2 def ObjectiveFunction.ObjectiveFunction.__str__ (self)

Human readable ObjectiveFunction print function.

Parameters

self	ObjectiveFunction pointer
	The ObjectiveFunction pointer.

9.4.3.3 def ObjectiveFunction.ObjectiveFunction.ackley (self, u)

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.
и	array
	The design parameters to be evaluated.

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)

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.
и	array
	The design parameters to be evaluated.

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)

Easom Function: Multimodal, n-dimensional.

Optimal example:

u = [pi, pi]

fitness = 1.0

Taken from: "Nature-Inspired Optimization Algorithms"

self	pointer
	The ObjectiveFunction pointer.

и	array
	The design parameters to be evaluated.

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)

Griewank Function: Multimodal, n-dimensional.

Optimal example:

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

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

pointer
The ObjectiveFunction pointer.
array
The design parameters to be evaluated.

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)

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.
и	array The design parameters to be evaluated. [x1, x2, x3, y1, y2, y3, y4]

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)

Mixed Integer Pressure vessel objective function.

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.
и	arrav
	The design parameters to be evaluated.

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)

Spring objective function.

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.
и	array
	The design parameters to be evaluated.

Returns

float: The fitness associated with the specified input.

9.4.3.10 def ObjectiveFunction.ObjectiveFunction.pressure_vessel (self, u)

Pressure vessel objective function.

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.
и	array The design parameters to be evaluated.

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)

Rastrigin Function: Multimodal, n-dimensional.

Optimal example: u = [0, 0, 0, ..., 0]

Taken from: "Nature-Inspired Optimization Algorithms"

Parameters

self	pointer The ObjectiveFunction pointer.
и	array

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)

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.
и	array

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)

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.
	The espective uncter pointer.
и	array The design parameters to be evaluated.
	The design parameters to se evaluated.

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)

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.
и	array The design parameters to be evaluated.

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)

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.
и	array The design parameters to be evaluated.

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)

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.
и	array
	The design parameters to be evaluated.

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)

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	
	The ObjectiveFunction pointer.
и	array

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)

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"

self	pointer
	The ObjectiveFunction pointer.

и	array

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)

Speed reducer objective function.

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.
и	array
	The design parameters to be evaluated.

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)

Spring objective function.

Optimal Example:

u = [0.05169046, 0.356750, 11.287126]

fitness = 0.0126653101469

Parameters

self	pointer The ObjectiveFunction pointer.
и	array The design parameters to be evaluated.

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)

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

Parameters

self	pointer
	The ObjectiveFunction pointer.
и	array

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)

Welded Beam objective function.

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.
и	array The design parameters to be evaluated.

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.

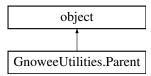
The documentation for this class was generated from the following file:

/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.

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.

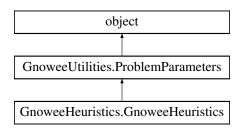
The documentation for this class was generated from the following file:

/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

Constructor for the ProblemParameters class.

def __repr__

ProblemParameters class attribute print function.

def __str__

Human readable ProblemParameters print function.

def sanitize_inputs

Checks and cleans user inputs to be compatible with expectations from the Gnowee algorithm.

• def map_to_discretes

Maps the sampled discrete indices to the array of allowable discrete values and returns the associated variable array.

· def map_from_discretes

Maps the discrete values to indices for sampling.

• def set_preset_params

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

Public Attributes

· objective

ObjectiveFunction Object: The objective function object to be used for the optimization.

constraints

list of Constraint Objects: The constraints on the optimization design space.

Ib

Checks and cleans user inputs to be compatible with expectations from the Gnowee algorithm.

• 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.

• cID

array: The continuous variable truth array.

• iID

array: The continuous variable truth array.

• dID

array: The continuous variable truth array.

• xID

array: The continuous variable truth array.

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, objective = None, constraints = [], lowerBounds = [], upperBounds = [], varType = [], discreteVals = [], optimum = 0.0, pltTitle = ", histTitle = ", varNames = ["])
```

Constructor for the ProblemParameters class.

The default constructor is useless for an optimization, but allows a placeholder class to be instantiated.

This class contains the problem definitions required for an optimization problem. It allows for single objective, multi-constraint mixed variable optimization and any subset thereof. At a minimum, the objective, lowerBounds, upperBounds, and varType attributes must be specified to run Gnowee.

The optimum is used for convergence criteria and can be input if known. If not, the default (0.0) will suffice for most problems, or the user can make an educated guess based on their knowledge of the problem.

Parameters

self	ProblemParameters pointer The ProblemParameters pointer.
objective	ObjectiveFunction object The optimization objective function to be used. Only a single objective function can be specified.
constraints	list of Constraint objects The constraints on the problem. Zero constraints can be specified as an empty list ([]), or multiple constraints can be specified as a list of Constraint objects.

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	Iist 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.map_from_discretes (self, variables)

Maps the discrete values to indices for sampling.

Parameters

self	pointer The ProblemParameters pointer. The Parent pointer.
variables	array The set of variables representing a design solution.

Returns

array: An array containing the variables associated with the design.

9.6.3.4 def GnoweeUtilities.ProblemParameters.map_to_discretes (self, variables)

Maps the sampled discrete indices to the array of allowable discrete values and returns the associated variable array.

Parameters

self	pointer
	The ProblemParameters pointer.
	The Parent pointer.
variables	array
	The set of variables representing a design solution.

Returns

array: An array containing the variables associated with the design.

9.6.3.5 def GnoweeUtilities.ProblemParameters.sanitize_inputs (self)

Checks and cleans user inputs to be compatible with expectations from the Gnowee algorithm.

Parameters

self	pointer
	The ProblemParameters pointer.

9.6.3.6 def GnoweeUtilities.ProblemParameters.set_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.

9.6.4 Member Data Documentation

9.6.4.1 def GnoweeUtilities.ProblemParameters.cID

array: The continuous variable truth array.

This contains a one in the positions corresponding to continuous variables and 0 otherwise.

9.6.4.2 GnoweeUtilities.ProblemParameters.constraints

list of Constraint Objects: The constraints on the optimization design space.

9.6.4.3 def GnoweeUtilities.ProblemParameters.dID

array: The continuous variable truth array.

This contains a one in the positions corresponding to continuous variables and 0 otherwise.

9.6.4.4 GnoweeUtilities.ProblemParameters.discreteVals

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

9.6.4.5 GnoweeUtilities.ProblemParameters.histTitle

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

9.6.4.6 def GnoweeUtilities.ProblemParameters.iID

array: The continuous variable truth array.

This contains a one in the positions corresponding to continuous variables and 0 otherwise.

9.6.4.7 GnoweeUtilities.ProblemParameters.lb

Checks and cleans user inputs to be compatible with expectations from the Gnowee algorithm.

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

Parameters

self	pointer
	The ProblemParameters pointer.

9.6.4.8 GnoweeUtilities.ProblemParameters.objective

ObjectiveFunction Object: The objective function object to be used for the optimization.

9.6.4.9 GnoweeUtilities.ProblemParameters.optimum

float: The global optimal solution.

9.6.4.10 GnoweeUtilities.ProblemParameters.pltTitle

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

9.6.4.11 GnoweeUtilities.ProblemParameters.ub

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

9.6.4.12 GnoweeUtilities.ProblemParameters.varNames

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

9.6.4.13 GnoweeUtilities.ProblemParameters.varType

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

9.6.4.14 def GnoweeUtilities.ProblemParameters.xID

array: The continuous variable truth array.

This contains a one in the positions corresponding to continuous variables and 0 otherwise.

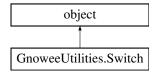
The documentation for this class was generated from the following file:

• /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__

Creates a switch class object to switch between cases.

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 Member Function Documentation

9.7.2.1 def GnoweeUtilities.Switch.__iter__ (self)

Return the match method once, then stop.

Parameters

self	pointer
	The Switch pointer.

9.7.2.2 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

9.7.3 Member Data Documentation

9.7.3.1 GnoweeUtilities.Switch.fall

boolean: Match indicator.

9.7.3.2 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 TSP.TSP Class Reference

This class creates a TSP object that can be used in optimization algorithms to solve the Travelling Saleman Problem.

Public Member Functions

```
    def __init__
        Constructor for the TSP class.
    def __repr__
        TSP class param print function.
    def __str__
        Human readable TSP print function.
```

def read_tsp

Read the starting TSP points from a TSPLIB standard file and populate class attributes.

def build_prob_params

Takes the current class attributes and populates a ProblemParameters object for use in optimization algorithms.

Public Attributes

name

string: The name of the TSPLIB problem.

· dimension

integer: The number of nodes (cities) in the problem.

nodes

list of lists: The coorinate pairs for each node.

• optimum

float: The optimal solution.

9.8.1 Detailed Description

This class creates a TSP object that can be used in optimization algorithms to solve the Travelling Saleman Problem.

9.8.2 Constructor & Destructor Documentation

```
9.8.2.1 def TSP.TSP.__init__ ( self, name = ", dimension = 1, nodes = [], optimum = 0.0)
```

Constructor for the TSP class.

Parameters

self	TSP pointer
	The TSP pointer.
name	string
	The name of the TSPLIB problem.
dimension	integer
	The number of nodes (cities) in the problem.
nodes	list of lists
	The coorinate pairs for each node.
optimum	float
	The optimal solution.

9.8.3 Member Function Documentation

9.8.3.1 def TSP.TSP.__repr__ (*self*)

TSP class param print function.

Parameters

self	TSP pointer
	The TSP pointer.

9.8.3.2 def TSP.TSP.__str__ (self)

Human readable TSP print function.

Parameters

self	TSP pointer
	The TSP pointer.

9.8.3.3 def TSP.TSP.build_prob_params (self, probParams)

Takes the current class attributes and populates a ProblemParameters object for use in optimization algorithms.

Parameters

probParams	ProblemParameters object
	A problem parameters object to be initialized with the class parameters.

9.8.3.4 def TSP.TSP.read_tsp (self, filename)

Read the starting TSP points from a TSPLIB standard file and populate class attributes.

Parameters

filename	string Path and filename of the tsp problem.

9.8.4 Member Data Documentation

9.8.4.1 TSP.TSP.dimension

integer: The number of nodes (cities) in the problem.

9.8.4.2 TSP.TSP.name

string: The name of the TSPLIB problem.

9.8.4.3 TSP.TSP.nodes

list of lists: The coorinate pairs for each node.

9.8.4.4 TSP.TSP.optimum

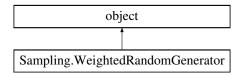
float: The optimal solution.

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

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

9.9 Sampling. Weighted Random Generator Class Reference

Defines a class of weights to be used to select based on linear weighting. Inheritance diagram for Sampling. Weighted Random Generator:



Public Member Functions

def __init__

WeightedRandomGenerator class constructor.

def next

Gets the next weight.

• def __call__

Gets the next weight.

Public Attributes

· totals

list or numpy array: The ordinal ranking or data that is used to generate tehe weights.

9.9.1 Detailed Description

Defines a class of weights to be used to select based on linear weighting.

This can be on index or some form of ordinal ranking.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 def Sampling.WeightedRandomGenerator.__init__ (self, weights)

WeightedRandomGenerator class constructor.

Parameters

self	pointer The WeightedRandomGenerator pointer.
weights	array The array of weights (Higher = more likely to be selected)

9.9.3 Member Function Documentation

9.9.3.1 def Sampling.WeightedRandomGenerator.__call__ (self)

Gets the next weight.

Parameters

self	pointer
	The WeightedRandomGenerator pointer.

Returns

integer: The randomly selected index of the weights array.

9.9.3.2 def Sampling.WeightedRandomGenerator.next (self)

Gets the next weight.

Parameters

self	pointer
	The WeightedRandomGenerator pointer.

Returns

integer: The randomly selected index of the weights array.

9.9.4 Member Data Documentation

$9.9.4.1 \quad Sampling. Weighted Random Generator. totals$

list or numpy array: The ordinal ranking or data that is used to generate tehe weights.

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

• /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Sampling.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.

76 File Documentation

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.

Namespaces

• Gnowee

Contains the Gnowee optimization program and associated utilities.

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

Classes

· class ObjectiveFunction.ObjectiveFunction

This 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

Classes

· class Sampling.WeightedRandomGenerator

Defines a class of weights to be used to select based on linear weighting.

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

78 File Documentation

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.

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

Classes

class TSP.TSP

This class creates a TSP object that can be used in optimization algorithms to solve the Travelling Saleman Problem.

Namespaces

Gnowee

Contains the Gnowee optimization program and associated utilities.

Index

/home/pyne-user/Dropbox/UCB/Research/ETAs/-	alpha
Design/Gnowee/src/Constraints.py, 75	GnoweeHeuristics::GnoweeHeuristics, 47
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	
Design/Gnowee/src/Gnowee.py, 75	build_prob_params
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	TSP::TSP, 71
Design/Gnowee/src/GnoweeHeuristics.py,	
75	cID
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	GnoweeUtilities::ProblemParameters, 67
Design/Gnowee/src/GnoweeUtilities.py, 76	changeCount
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	GnoweeUtilities::Parent, 62
Design/Gnowee/src/ObjectiveFunction.py, 76	constraint
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	Constraints::Constraint, 36
Design/Gnowee/src/OptiPlot.py, 77	Constraints, 13
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	constraints
Design/Gnowee/src/Sampling.py, 77	GnoweeUtilities::ProblemParameters, 67
/home/pyne-user/Dropbox/UCB/Research/ETAs/-	Constraints.Constraint, 29
Design/Gnowee/src/TSP.py, 78	Constraints::Constraint
call	init, 30
Sampling::WeightedRandomGenerator, 73	repr , 30
init	str, 31
Constraints::Constraint, 30	constraint, 36
GnoweeHeuristics::GnoweeHeuristics, 39	func, 36
GnoweeUtilities, 17	get_penalty, 31
GnoweeUtilities::Event, 37	greater_than, 31
GnoweeUtilities::Parent, 61	less or equal, 31
GnoweeUtilities::ProblemParameters, 64	less_than, 32
ObjectiveFunction::ObjectiveFunction, 50	mi_chemical_process, 32
Sampling::WeightedRandomGenerator, 73	mi_pressure_vessel, 32
TSP::TSP, 70	mi_spring, 33
iter	pressure_vessel, 33
GnoweeUtilities::Switch, 69	set_constraint_func, 34
repr	speed_reducer, 34
Constraints::Constraint, 30	spring, 35
GnoweeHeuristics::GnoweeHeuristics, 41	welded_beam, 35
GnoweeUtilities::Event, 37	cont_levy_flight
GnoweeUtilities::Parent, 62	GnoweeHeuristics::GnoweeHeuristics, 41
GnoweeUtilities::ProblemParameters, 65	convTol
ObjectiveFunction::ObjectiveFunction, 50	GnoweeHeuristics::GnoweeHeuristics, 47
TSP::TSP, 71	crossover
	GnoweeHeuristics::GnoweeHeuristics, 41
str Constraints::Constraint, 31	anoweer learnsticsanoweer learnstics, 41
GnoweeHeuristics::GnoweeHeuristics, 41	dID
GnoweeUtilities::Event, 37	GnoweeUtilities::ProblemParameters, 67
GnoweeUtilities::Parent, 62	dejong
	ObjectiveFunction::ObjectiveFunction, 52
GnoweeUtilities::ProblemParameters, 65	design
ObjectiveFunction::ObjectiveFunction, 50	GnoweeUtilities::Event, 37
TSP::TSP, 71	dimension
ackley	TSP::TSP, 72
ackley Objective Function: Objective Function, 50	•
ObjectiveFunction::ObjectiveFunction, 50	disc_levy_flight

80 INDEX

GnoweeHeuristics::GnoweeHeuristics, 43	n, 47
discreteVals	optConvTol, 48
GnoweeUtilities::ProblemParameters, 67	penalty, 48
	population, 48
easom	population_update, 45
ObjectiveFunction::ObjectiveFunction, 52	scalingFactor, 48
elite_crossover	scatter_search, 46
GnoweeHeuristics::GnoweeHeuristics, 43	stallLimit, 48
evaluations	GnoweeUtilities, 17
GnoweeUtilities::Event, 37	init, 17
	GnoweeUtilities.Event, 36
fall	GnoweeUtilities.Parent, 61
GnoweeUtilities::Switch, 69	GnoweeUtilities.ProblemParameters, 63
fitness	GnoweeUtilities.Switch, 68
GnoweeUtilities::Event, 37	GnoweeUtilities::Event
GnoweeUtilities::Parent, 62	init, 37
fracDiscovered	, 37 repr, 37
GnoweeHeuristics::GnoweeHeuristics, 47	, 37 str, 37
fracElite	
GnoweeHeuristics::GnoweeHeuristics, 47	design, 37
fracLevy	evaluations, 37
GnoweeHeuristics::GnoweeHeuristics, 47	fitness, 37
func	generation, 38
Constraints::Constraint, 36	GnoweeUtilities::Parent
ObjectiveFunction::ObjectiveFunction, 60	init, 61
•	repr, 62
gamma	str, 62
GnoweeHeuristics::GnoweeHeuristics, 47	changeCount, 62
generation	fitness, 62
GnoweeUtilities::Event, 38	stallCount, 62
get_cdr_permutations	variables, 62
Sampling, 22	GnoweeUtilities::ProblemParameters
get_penalty	init, 64
Constraints::Constraint, 31	repr, 65
Gnowee, 14, 27	str, 65
main, 14	cID, 67
GnoweeHeuristics, 15	constraints, 67
rejection_bounds, 15	dID, 67
simple_bounds, 16	discreteVals, 67
GnoweeHeuristics.GnoweeHeuristics, 38	histTitle, 67
	iID, 67
GnoweeHeuristics::GnoweeHeuristics	lb, 67
init, 39	map from discretes, 66
repr, 41	map_to_discretes, 66
str, 41	objective, 68
alpha, 47	
cont_levy_flight, 41	optimum, 68
convTol, 47	pltTitle, 68
crossover, 41	sanitize_inputs, 66
disc_levy_flight, 43	set_preset_params, 66
elite_crossover, 43	ub, 68
fracDiscovered, 47	varNames, 68
fracElite, 47	varType, 68
fracLevy, 47	xID, 68
gamma, 47	GnoweeUtilities::Switch
initSampling, 47	iter, 69
initialize, 43	fall, 69
maxFevals, 47	match, 69
maxGens, 47	value, 69
mutate, 45	greater_than
,	-

INDEX 81

Constraints::Constraint, 31 griewank	objective GnoweeUtilities::ProblemParameters, 68
ObjectiveFunction::ObjectiveFunction, 53	ObjectiveFunction::ObjectiveFunction, 60
Objective unctionObjective unction, 33	Objective function, 18
histTitle	prod, 18
GnoweeUtilities::ProblemParameters, 67	ObjectiveFunction.ObjectiveFunction, 48
drioweed linites Toblethi drameters, 07	ObjectiveFunction::ObjectiveFunction
iID	init, 50
GnoweeUtilities::ProblemParameters, 67	repr, 50
initSampling	
GnoweeHeuristics::GnoweeHeuristics, 47	ackley, 50
initial samples	dejong, 52
Sampling, 23	easom, 52
initialize	func, 60
GnoweeHeuristics::GnoweeHeuristics, 43	griewank, 53
,	mi_chemical_process, 53
lb	mi_pressure_vessel, 53
GnoweeUtilities::ProblemParameters, 67	mi_spring, 54
less_or_equal	objective, 60
Constraints::Constraint, 31	pressure_vessel, 54
less_than	rastrigin, 55
Constraints::Constraint, 32	rosenbrock, 55
levy	set_obj_func, 56
Sampling, 23	shifted_ackley, 56
	shifted_dejong, 56
main	shifted_easom, 57
Gnowee, 14	shifted_griewank, 57
map_from_discretes	shifted_rastrigin, 58
GnoweeUtilities::ProblemParameters, 66	shifted_rosenbrock, 58
map_to_discretes	speed_reducer, 59
GnoweeUtilities::ProblemParameters, 66	spring, 59
match	tsp, 59
GnoweeUtilities::Switch, 69	welded_beam, 60
maxFevals	optConvToI
GnoweeHeuristics::GnoweeHeuristics, 47	GnoweeHeuristics::GnoweeHeuristics, 48
maxGens	OptiPlot, 19
GnoweeHeuristics::GnoweeHeuristics, 47	plot_feval_hist, 19
mi_chemical_process	plot_hist, 20
Constraints::Constraint, 32	plot_hist_comp, 20
ObjectiveFunction::ObjectiveFunction, 53	plot optimization, 20
mi_pressure_vessel	plot_tlf, 21
Constraints::Constraint, 32	plot_vars, 21
ObjectiveFunction::ObjectiveFunction, 53	optimum
mi_spring	GnoweeUtilities::ProblemParameters, 68
Constraints::Constraint, 33	TSP::TSP, 72
ObjectiveFunction::ObjectiveFunction, 54	,
mutate	params
GnoweeHeuristics::GnoweeHeuristics, 45	Sampling, 24
	penalty
n	GnoweeHeuristics::GnoweeHeuristics, 48
GnoweeHeuristics::GnoweeHeuristics, 47	plot_feval_hist
NOLH	OptiPlot, 19
Sampling, 24	plot_hist
name	OptiPlot, 20
TSP::TSP, 72	plot_hist_comp
next	OptiPlot, 20
Sampling::WeightedRandomGenerator, 73	plot_optimization
nodes	OptiPlot, 20
TSP::TSP, 72	plot_samples

82 INDEX

Sampling, 24	ObjectiveFunction::ObjectiveFunction, 57
plot_tlf	shifted_rastrigin
OptiPlot, 21	ObjectiveFunction::ObjectiveFunction, 58
plot_vars	shifted_rosenbrock
OptiPlot, 21	ObjectiveFunction::ObjectiveFunction, 58
pltTitle	simple_bounds
GnoweeUtilities::ProblemParameters, 68	GnoweeHeuristics, 16
population	speed_reducer
GnoweeHeuristics::GnoweeHeuristics, 48	Constraints::Constraint, 34
population_update	ObjectiveFunction::ObjectiveFunction, 59
GnoweeHeuristics::GnoweeHeuristics, 45	spring
pressure_vessel	Constraints::Constraint, 35
Constraints::Constraint, 33	ObjectiveFunction::ObjectiveFunction, 59
ObjectiveFunction::ObjectiveFunction, 54	stallCount
prod	GnoweeUtilities::Parent, 62
ObjectiveFunction, 18	stallLimit
	GnoweeHeuristics::GnoweeHeuristics, 48
rastrigin	TOP
ObjectiveFunction::ObjectiveFunction, 55	TSP, 26
read_tsp	TSP.TSP, 70
TSP::TSP, 71	TSP::TSP
rejection_bounds	init, 70
GnoweeHeuristics, 15	repr, 71
rosenbrock	str, 71
ObjectiveFunction::ObjectiveFunction, 55	build_prob_params, 71
	dimension, 72
Sampling, 22	name, 72
get_cdr_permutations, 22	nodes, 72
initial_samples, 23	optimum, 72
levy, 23	read_tsp, 71
NOLH, 24	tlf
params, 24	Sampling, 25
plot_samples, 24	totals
tlf, 25	Sampling::WeightedRandomGenerator, 74
Sampling.WeightedRandomGenerator, 72	tsp
Sampling::WeightedRandomGenerator	ObjectiveFunction::ObjectiveFunction, 59
call, 73	
init, 73	ub
next, 73	GnoweeUtilities::ProblemParameters, 68
totals, 74	value
sanitize_inputs	GnoweeUtilities::Switch, 69
GnoweeUtilities::ProblemParameters, 66	varNames
scalingFactor	
GnoweeHeuristics::GnoweeHeuristics, 48	GnoweeUtilities::ProblemParameters, 68
scatter_search	varType
GnoweeHeuristics::GnoweeHeuristics, 46	GnoweeUtilities::ProblemParameters, 68 variables
set_constraint_func	
Constraints::Constraint, 34	GnoweeUtilities::Parent, 62
set_obj_func	welded_beam
ObjectiveFunction::ObjectiveFunction, 56	Constraints::Constraint, 35
set_preset_params	ObjectiveFunction::ObjectiveFunction, 60
GnoweeUtilities::ProblemParameters, 66	Objective runctionObjective runction, 80
shifted_ackley	xID
ObjectiveFunction::ObjectiveFunction, 56	GnoweeUtilities::ProblemParameters, 68
shifted_dejong	anomocountico robiciiii arameters, 00
ObjectiveFunction::ObjectiveFunction, 56	
shifted_easom	
ObjectiveFunction::ObjectiveFunction, 57	
shifted_griewank	