

Gnowee

Generated by Doxygen 1.8.6

Tue May 16 2017 15:39:13



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Namespace Index</b>	<b>5</b>
3.1	Packages . . . . .	5
<b>4</b>	<b>Hierarchical Index</b>	<b>7</b>
4.1	Class Hierarchy . . . . .	7
<b>5</b>	<b>Class Index</b>	<b>9</b>
5.1	Class List . . . . .	9
<b>6</b>	<b>File Index</b>	<b>11</b>
6.1	File List . . . . .	11
<b>7</b>	<b>Module Documentation</b>	<b>13</b>
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	contains_sublist . . . . .	15
7.3.2.2	rejection_bounds . . . . .	16
7.3.2.3	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	ObjectiveFunction	18
7.5.1	Detailed Description	18
7.5.2	Function Documentation	18
7.5.2.1	prod	18
7.6	OptiPlot	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	Sampling	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
<b>8</b>	<b>Namespace Documentation</b>	<b>27</b>
8.1	Gnowee Namespace Reference	27
8.1.1	Detailed Description	27
<b>9</b>	<b>Class Documentation</b>	<b>29</b>
9.1	Constraints.Constraint Class Reference	29
9.1.1	Detailed Description	30
9.1.2	Constructor & Destructor Documentation	30
9.1.2.1	__init__	30
9.1.3	Member Function Documentation	30
9.1.3.1	__repr__	30
9.1.3.2	__str__	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	<a href="#">less_than</a>	32
9.1.3.7	<a href="#">mi_chemical_process</a>	32
9.1.3.8	<a href="#">mi_pressure_vessel</a>	32
9.1.3.9	<a href="#">mi_spring</a>	33
9.1.3.10	<a href="#">pressure_vessel</a>	33
9.1.3.11	<a href="#">set_constraint_func</a>	34
9.1.3.12	<a href="#">speed_reducer</a>	34
9.1.3.13	<a href="#">spring</a>	35
9.1.3.14	<a href="#">welded_beam</a>	35
9.1.4	<a href="#">Member Data Documentation</a>	36
9.1.4.1	<a href="#">constraint</a>	36
9.1.4.2	<a href="#">func</a>	36
9.2	<a href="#">GnoweeUtilities.Event Class Reference</a>	36
9.2.1	<a href="#">Detailed Description</a>	36
9.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	37
9.2.2.1	<a href="#">__init__</a>	37
9.2.3	<a href="#">Member Function Documentation</a>	37
9.2.3.1	<a href="#">__repr__</a>	37
9.2.3.2	<a href="#">__str__</a>	37
9.2.4	<a href="#">Member Data Documentation</a>	37
9.2.4.1	<a href="#">design</a>	37
9.2.4.2	<a href="#">evaluations</a>	37
9.2.4.3	<a href="#">fitness</a>	38
9.2.4.4	<a href="#">generation</a>	38
9.3	<a href="#">GnoweeHeuristics.GnoweeHeuristics Class Reference</a>	38
9.3.1	<a href="#">Detailed Description</a>	39
9.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
9.3.2.1	<a href="#">__init__</a>	40
9.3.3	<a href="#">Member Function Documentation</a>	41
9.3.3.1	<a href="#">__repr__</a>	41
9.3.3.2	<a href="#">__str__</a>	41
9.3.3.3	<a href="#">comb_levy_flight</a>	41
9.3.3.4	<a href="#">cont_levy_flight</a>	42
9.3.3.5	<a href="#">crossover</a>	42
9.3.3.6	<a href="#">disc_levy_flight</a>	43
9.3.3.7	<a href="#">elite_crossover</a>	43
9.3.3.8	<a href="#">initialize</a>	43
9.3.3.9	<a href="#">mutate</a>	44
9.3.3.10	<a href="#">population_update</a>	44
9.3.3.11	<a href="#">scatter_search</a>	45

9.3.3.12	<a href="#">three_opt</a>	46
9.3.3.13	<a href="#">two_opt</a>	46
9.3.4	<a href="#">Member Data Documentation</a>	46
9.3.4.1	<a href="#">alpha</a>	46
9.3.4.2	<a href="#">convTol</a>	46
9.3.4.3	<a href="#">fracDiscovered</a>	46
9.3.4.4	<a href="#">fracElite</a>	46
9.3.4.5	<a href="#">fracLevy</a>	47
9.3.4.6	<a href="#">gamma</a>	47
9.3.4.7	<a href="#">initSampling</a>	47
9.3.4.8	<a href="#">maxFevals</a>	47
9.3.4.9	<a href="#">maxGens</a>	47
9.3.4.10	<a href="#">n</a>	47
9.3.4.11	<a href="#">optConvTol</a>	47
9.3.4.12	<a href="#">penalty</a>	47
9.3.4.13	<a href="#">population</a>	47
9.3.4.14	<a href="#">scalingFactor</a>	47
9.3.4.15	<a href="#">stallLimit</a>	47
9.4	<a href="#">ObjectiveFunction.ObjectiveFunction Class Reference</a>	48
9.4.1	<a href="#">Detailed Description</a>	49
9.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	49
9.4.2.1	<a href="#">__init__</a>	49
9.4.3	<a href="#">Member Function Documentation</a>	49
9.4.3.1	<a href="#">__repr__</a>	49
9.4.3.2	<a href="#">__str__</a>	50
9.4.3.3	<a href="#">ackley</a>	50
9.4.3.4	<a href="#">dejong</a>	50
9.4.3.5	<a href="#">easom</a>	51
9.4.3.6	<a href="#">griewank</a>	51
9.4.3.7	<a href="#">mi_chemical_process</a>	51
9.4.3.8	<a href="#">mi_pressure_vessel</a>	52
9.4.3.9	<a href="#">mi_spring</a>	52
9.4.3.10	<a href="#">pressure_vessel</a>	53
9.4.3.11	<a href="#">rastrigin</a>	53
9.4.3.12	<a href="#">rosenbrock</a>	54
9.4.3.13	<a href="#">set_obj_func</a>	54
9.4.3.14	<a href="#">shifted_ackley</a>	54
9.4.3.15	<a href="#">shifted_dejong</a>	55
9.4.3.16	<a href="#">shifted_easom</a>	55
9.4.3.17	<a href="#">shifted_griewank</a>	56

9.4.3.18	shifted_rastrigin	56
9.4.3.19	shifted_rosenbrock	56
9.4.3.20	speed_reducer	57
9.4.3.21	spring	57
9.4.3.22	tsp	58
9.4.3.23	welded_beam	58
9.4.4	Member Data Documentation	58
9.4.4.1	func	59
9.4.4.2	objective	59
9.5	GnoweeUtilities.Parent Class Reference	59
9.5.1	Detailed Description	59
9.5.2	Constructor & Destructor Documentation	60
9.5.2.1	__init__	60
9.5.3	Member Function Documentation	61
9.5.3.1	__repr__	61
9.5.3.2	__str__	61
9.5.4	Member Data Documentation	61
9.5.4.1	changeCount	61
9.5.4.2	fitness	61
9.5.4.3	stallCount	61
9.5.4.4	variables	62
9.6	GnoweeUtilities.ProblemParameters Class Reference	62
9.6.1	Detailed Description	63
9.6.2	Constructor & Destructor Documentation	63
9.6.2.1	__init__	63
9.6.3	Member Function Documentation	64
9.6.3.1	__repr__	64
9.6.3.2	__str__	65
9.6.3.3	map_from_discretes	65
9.6.3.4	map_to_discretes	65
9.6.3.5	sanitize_inputs	65
9.6.3.6	set_preset_params	66
9.6.4	Member Data Documentation	66
9.6.4.1	cID	66
9.6.4.2	constraints	66
9.6.4.3	dID	66
9.6.4.4	discreteVals	66
9.6.4.5	histTitle	67
9.6.4.6	iID	67
9.6.4.7	lb	67

9.6.4.8	<a href="#">objective</a>	67
9.6.4.9	<a href="#">optimum</a>	67
9.6.4.10	<a href="#">pltTitle</a>	67
9.6.4.11	<a href="#">ub</a>	67
9.6.4.12	<a href="#">varNames</a>	67
9.6.4.13	<a href="#">varType</a>	67
9.6.4.14	<a href="#">xID</a>	67
9.7	<a href="#">GnoweeUtilities.Switch Class Reference</a>	68
9.7.1	<a href="#">Detailed Description</a>	68
9.7.2	<a href="#">Member Function Documentation</a>	68
9.7.2.1	<a href="#">__iter__</a>	68
9.7.2.2	<a href="#">match</a>	68
9.7.3	<a href="#">Member Data Documentation</a>	69
9.7.3.1	<a href="#">fall</a>	69
9.7.3.2	<a href="#">value</a>	69
9.8	<a href="#">TSP.TSP Class Reference</a>	69
9.8.1	<a href="#">Detailed Description</a>	70
9.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	70
9.8.2.1	<a href="#">__init__</a>	70
9.8.3	<a href="#">Member Function Documentation</a>	70
9.8.3.1	<a href="#">__repr__</a>	70
9.8.3.2	<a href="#">__str__</a>	70
9.8.3.3	<a href="#">build_prob_params</a>	71
9.8.3.4	<a href="#">read_tsp</a>	71
9.8.4	<a href="#">Member Data Documentation</a>	71
9.8.4.1	<a href="#">dimension</a>	71
9.8.4.2	<a href="#">name</a>	71
9.8.4.3	<a href="#">nodes</a>	71
9.8.4.4	<a href="#">optimum</a>	71
9.9	<a href="#">Sampling.WeightedRandomGenerator Class Reference</a>	71
9.9.1	<a href="#">Detailed Description</a>	72
9.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	72
9.9.2.1	<a href="#">__init__</a>	72
9.9.3	<a href="#">Member Function Documentation</a>	72
9.9.3.1	<a href="#">__call__</a>	72
9.9.3.2	<a href="#">next</a>	73
9.9.4	<a href="#">Member Data Documentation</a>	73
9.9.4.1	<a href="#">totals</a>	73



10.1	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Constraints.py File Reference	75
10.2	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Gnowee.py File Reference . .	75
10.3	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeHeuristics.py File Reference . . . . .	75
10.4	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/GnoweeUtilities.py File Reference . . . . .	76
10.5	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ObjectiveFunction.py File Reference . . . . .	76
10.6	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/OptiPlot.py File Reference . .	77
10.7	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/Sampling.py File Reference .	77
10.8	/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/TSP.py File Reference . . . .	78
<b>Index</b>		<b>79</b>



# Chapter 1

## Main Page

### Gnowee

Version

1.0

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

### Running [Gnowee](#)

For examples on how to run [Gnowee](#), please refer to the runGnowee notebook included in the src directory.

### Building Documentation

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

```
>> doxygen Doxyfile
```

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

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

```
>> make
```

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

### Citation Information

To cite [Gnowee](#), use the following reference:

### Contact information

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

Alternatively, questions or comments on [Gnowee](#) can be directed to:

James Bevins

james.e.bevins@gmail.com

## Licensing Information

## Acknowledgements

AF, advisor, NSF

## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Constraints . . . . .	13
Gnowee . . . . .	14
GnoweeHeuristics . . . . .	15
GnoweeUtilities . . . . .	17
ObjectiveFunction . . . . .	18
OptiPlot . . . . .	19
Sampling . . . . .	22
TSP . . . . .	26



## Chapter 3

# Namespace Index

### 3.1 Packages

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

<a href="#">Gnowee</a>	Contains the <a href="#">Gnowee</a> optimization program and associated utilities . . . . .	<a href="#">27</a>
------------------------	---------------------------------------------------------------------------------------------	--------------------





## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

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

object	
Constraints.Constraint . . . . .	29
GnoweeUtilities.Event . . . . .	36
GnoweeUtilities.Parent . . . . .	59
GnoweeUtilities.ProblemParameters . . . . .	62
GnoweeHeuristics.GnoweeHeuristics . . . . .	38
GnoweeUtilities.Switch . . . . .	68
ObjectiveFunction.ObjectiveFunction . . . . .	48
Sampling.WeightedRandomGenerator . . . . .	71
TSP.TSP . . . . .	69



## Chapter 5

# Class Index

### 5.1 Class List

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

<a href="#">Constraints.Constraint</a>	
The class creates a Constraints object that can be used in optimization algorithms . . . . .	29
<a href="#">GnoweeUtilities.Event</a>	
Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback . . . . .	36
<a href="#">GnoweeHeuristics.GnoweeHeuristics</a>	
The class is the foundation of the <a href="#">Gnowee</a> optimization algorithm . . . . .	38
<a href="#">ObjectiveFunction.ObjectiveFunction</a>	
This class creates a <a href="#">ObjectiveFunction</a> object that can be used in optimization algorithms . . .	48
<a href="#">GnoweeUtilities.Parent</a>	
The class contains all of the parameters pertinent to a member of the population . . . . .	59
<a href="#">GnoweeUtilities.ProblemParameters</a>	
Creates an object containing key features of the chosen optimization problem . . . . .	62
<a href="#">GnoweeUtilities.Switch</a>	
Creates a switch class object to switch between cases . . . . .	68
<a href="#">TSP.TSP</a>	
This class creates a <a href="#">TSP</a> object that can be used in optimization algorithms to solve the Travelling Saleman Problem . . . . .	69
<a href="#">Sampling.WeightedRandomGenerator</a>	
Defines a class of weights to be used to select based on linear weighting . . . . .	71



## Chapter 6

# 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/ <a href="#">Constraints.py</a> . . . . .	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">Gnowee.py</a> . . . . .	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">GnoweeHeuristics.py</a> . . . . .	75
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">GnoweeUtilities.py</a> . . . . .	76
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">ObjectiveFunction.py</a> . . . . .	76
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">OptiPlot.py</a> . . . . .	77
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">Sampling.py</a> . . . . .	77
/home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/ <a href="#">TSP.py</a> . . . . .	78



# Chapter 7

## Module Documentation

### 7.1 Constraints

Defines a class to perform constraint calculations.

#### Classes

- class [Constraints.Constraint](#)

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

#### 7.1.1 Detailed Description

Defines a class to perform constraint calculations.

#### Author

James Bevins,

#### Date

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

16May17

#### 7.2.2 Function Documentation

##### 7.2.2.1 `def Gnowee.main ( gh )`

Main controller program for the [Gnowee](#) optimization.

#### Parameters

<i>gh</i>	<i>GnoweeHeuristic object</i> An object constaining the problem definition and the settings and methods required for the <a href="#">Gnowee</a> 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

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.
- def [GnoweeHeuristics.contains\\_sublist](#)  
Find index of sublist, if it exists.

#### 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

16May17

#### 7.3.2 Function Documentation

##### 7.3.2.1 def GnoweeHeuristics.contains\_sublist ( lst, sublist )

Find index of sublist, if it exists.

#### Parameters

<i>lst</i>	<i>list</i> The list in which to search for sublist.
<i>sublst</i>	<i>list</i> The list to search for.

#### Returns

*integer*: Index location of sublist in lst.

### 7.3.2.2 `def GnoweeHeuristics.rejection_bounds ( parent, child, stepSize, lb, ub )`

Application of problem boundaries to generated solutions.

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

#### Parameters

<i>parent</i>	<i>array</i> The current system designs.
<i>child</i>	<i>array</i> The proposed new system designs.
<i>stepSize</i>	<i>float</i> The stepsize for the permutation.
<i>lb</i>	<i>array</i> The lower bounds of the design variable(s).
<i>ub</i>	<i>array</i> The upper bounds of the design variable(s).

#### Returns

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

### 7.3.2.3 `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

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

#### Returns

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

## 7.4 GnoweeUtilities

Classes and methods to support the [Gnowee](#) optimization algorithm.

### Classes

- class [GnoweeUtilities.Parent](#)  
*The class contains all of the parameters pertinent to a member of the population.*
- class [GnoweeUtilities.Event](#)  
*Represents a snapshot in the optimization process to be used for debugging, benchmarking, and user feedback.*
- class [GnoweeUtilities.ProblemParameters](#)  
*Creates an object containing key features of the chosen optimization problem.*
- class [GnoweeUtilities.Switch](#)  
*Creates a switch class object to switch between cases.*

### 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

15May17

#### 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.

##### Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Switch</a> pointer.
<i>value</i>	<i>string</i> 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, multiplicative 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, multiplicative equivalent to sum).

#### Parameters

<i>iterable</i>	<i>list or array or generator</i>	Iterable set to multiply.
-----------------	-----------------------------------	---------------------------

#### Returns

*float*: The product of all of the items in iterable

## 7.6 OptiPlot

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

### Functions

- def [OptiPlot.plot\\_vars](#)  
*Plot the variables as they change in the optimization process.*
- def [OptiPlot.plot\\_hist](#)  
*Plots the histogram of function evaluation results from multiple runs of an optimization algorithm.*
- def [OptiPlot.plot\\_hist\\_comp](#)  
*Histograms and plots the comparison of two sets of function evaluation data.*
- def [OptiPlot.plot\\_feval\\_hist](#)  
*Plots the fitness vs function evaluation results of an optimization algorithm run.*
- def [OptiPlot.plot\\_tlf](#)  
*Plots a comparison of the TLF to the Levy distribution.*
- def [OptiPlot.plot\\_optimization](#)  
*Plots the results of optimization process for a given algorithm and parameter.*

### 7.6.1 Detailed Description

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

#### Author

James Bevins

#### Date

10May17

### 7.6.2 Function Documentation

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

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

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

#### Parameters

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

<i>label</i>	<i>list</i> 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

<i>data</i>	<i>list</i> Contains the number of function evals for each optimization run.
<i>title</i>	<i>string</i> Title for plot.
<i>xLabel</i>	<i>string</i> 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

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

#### 7.6.2.4 `def OptiPlot.plot_optimization ( data, label, title = "" )`

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

##### Parameters

<i>data</i>	<i>array</i> Contains the function eval history. Columns are: [function evals, fitness, number of datapoints]
-------------	------------------------------------------------------------------------------------------------------------------

<i>label</i>	<i>list</i> List of names of the problem types ran.
<i>title</i>	<i>string</i> 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

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

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

Plot the variables as they change in the optimization process.

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

Parameters

<i>data</i>	<i>list of event objects</i> Contain the optimization history in event objects within the data list.
<i>lowBounds</i>	<i>array</i> The lower bounds of the design variable(s).
<i>upBounds</i>	<i>array</i> The upper bounds of the design variable(s).
<i>title</i>	<i>string</i> Title for plot.
<i>label</i>	<i>list</i> 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 permutations 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

15May17

#### 7.7.2 Function Documentation

##### 7.7.2.1 def Sampling.get\_cdr\_permutations ( dim )

Generate a set of CDR permutations for NOLH.



## Parameters

<i>dim</i>	<i>integer</i> The dimension of the space.
------------	-----------------------------------------------

## Returns

*array*: A configuration vector.

*array*: Array containing the indexes of the columns 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', 'lhc', or 'rand-wor'.

## Parameters

<i>lb</i>	<i>array</i> The lower bounds of the design variable(s).
<i>ub</i>	<i>array</i> The upper bounds of the design variable(s).
<i>method</i>	<i>string</i> String representing the chosen sampling method. Valid options are: 'random', 'nolh', 'nolh-rp', 'nolh-cdr', 'lhc', 'random-wor'.
<i>numSamp</i>	<i>integer</i> 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.

$$L_{\alpha,\gamma}(z) = \frac{1}{\pi} \int_0^{+\infty} e^{-\gamma q^\alpha} \cos(qz) dq$$

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

## Parameters

<i>nc</i>	<i>integer</i> The number of columns of Levy values for the return array.
<i>nr</i>	<i>integer</i> The number of rows of Levy values for the return array.

<i>alpha</i>	<i>float</i> Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
<i>gam</i>	<i>float</i> Gamma - Scale unit of process for Levy flights.
<i>n</i>	<i>integer</i> Number of independent variables - can be used to reduce Levy flight sampling variance.

#### Returns

*array*: Array representing the levy flights for each nest.

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

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

(2012) and reference therein.

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

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

$d = m + \{m-1\}_2$

is the NOLH dimensionality.

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

#### Parameters

<i>conf</i>	<i>array</i> Configuration vector.
<i>remove</i>	<i>array</i> Array containing the indexes of the columns to be removed from conf vector.

#### Returns

*array*: Array containing nearly orthogonal latin hypercube sampling.

#### 7.7.2.5 def Sampling.params ( *dim* )

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

#### Parameters

<i>dim</i>	<i>integer</i> The dimension of the space.
------------	-----------------------------------------------

7.7.2.6 `def Sampling.plot_samples ( s )`

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

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

## Parameters

<i>s</i>	<i>array</i> The list of coordinates for the sampled phase space.
----------	----------------------------------------------------------------------

7.7.2.7 `def Sampling.tlf ( numRows = 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

<i>numRow</i>	<i>integer</i> Number of rows of Levy flights to sample.
<i>numCol</i>	<i>integer</i> Number of columns of Levy flights to sample.
<i>alpha</i>	<i>float</i> Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
<i>gam</i>	<i>float</i> Gamma - Scale unit of process for Levy flights.
<i>cutPoint</i>	<i>float</i> 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

15May17

## 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](#)



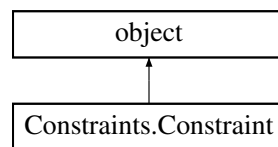
## Chapter 9

# 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 specified maximum including that maximum.*
- `def less_than`  
*Compares a previously calculated value to a user specified 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

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

### 9.1.3 Member Function Documentation

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

`Constraint` class param print function.

#### Parameters



<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
-------------	-----------------------------------------------------------

#### 9.1.3.2 def Constraints.Constraint.\_\_str\_\_( self )

Human readable [Constraint](#) print function.

Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
-------------	-----------------------------------------------------------

#### 9.1.3.3 def Constraints.Constraint.get\_penalty( self, violation )

Calculate the constraint violation penalty, if any.

Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
<i>violation</i>	<i>float</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
<i>candidate</i>	<i>float</i> 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 specified maximum including that maximum.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
<i>candidate</i>	<i>float</i> 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 specified maximum excluding that maximum.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
<i>candidate</i>	<i>float</i> 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

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">Constraint</a> pointer.
<i>funcName</i>	<i>string</i> A string identifying the constraint function to be used.

#### 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"

#### Parameters

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
-------------	--------------------------------------------------

<i>u</i>	<i>array</i> 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**

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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**

<i>self</i>	<i>pointer</i> The ObjectiveFunction pointer.
<i>u</i>	<i>array</i> 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.

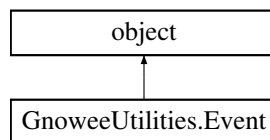
The documentation for this class was generated from the following file:

- </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

<i>self</i>	<a href="#">Event</a> pointer The <a href="#">Event</a> pointer.
<i>generation</i>	<i>integer</i> The generation the design was arrived at.
<i>evaluations</i>	<i>integer</i> The number of fitness evaluations done to obtain this design.
<i>fitness</i>	<i>float</i> The assessed fitness for the current set of variables.
<i>design</i>	<i>array</i> 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

<i>self</i>	<a href="#">Event</a> pointer The <a href="#">Event</a> pointer.
-------------	---------------------------------------------------------------------

### 9.2.3.2 def GnoweeUtilities.Event.\_\_str\_\_( self )

Human readable [Event](#) print function.

Parameters

<i>self</i>	<a href="#">Event</a> pointer The <a href="#">Event</a> 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.

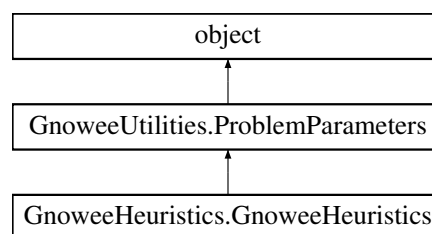
The documentation for this class was generated from the following file:

- /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 [comb\\_levy\\_flight](#)  
Generate new children using truncated Levy flights permutation and inversion of current generation design parameters.
- 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 [two\\_opt](#)  
*Generate new children using the two\_opt operator.*
- def [three\\_opt](#)  
*Generate new children using the three\_opt operator.*
- 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 generations to search without a decrease exceeding convTol.*
- [optConvTol](#)  
*float: The maximum deviation from the best know fitness value before the search terminates.*
- [xID](#)

### 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 heuristics.

### 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 = 10000, 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 inherited 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="">](#).

#### Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>population</i>	<i>integer</i> The number of members in each generation.
<i>initSampling</i>	<i>string</i> 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 <code>init_samples()</code> .
<i>fracDiscovered</i>	<i>: float</i> Discovery probability used for the <a href="#">mutate()</a> heuristic.
<i>fracElite</i>	<i>float</i> Elite fraction probability used for the <a href="#">scatter_search()</a> , <a href="#">crossover()</a> , and <code>cont_crossover()</code> heuristics.
<i>fracLevy</i>	<i>float</i> Levy flight probability used for the <a href="#">disc_levy_flight()</a> and <a href="#">cont_levy_flight()</a> heuristics.
<i>alpha</i>	<i>float</i> Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process.
<i>gamma</i>	<i>float</i> Gamma - scale unit of process for Levy flights.
<i>n</i>	<i>integer</i> Number of independent variables - can be used to reduce Levy flight sampling variance.
<i>penalty</i>	<i>float</i> Individual constraint violation penalty to add to objective function.

<i>scalingFactor</i>	<i>float</i> 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.
<i>maxGens</i>	<i>integer</i> The maximum number of generations to search.
<i>maxFevals</i>	<i>integer</i> The maximum number of objective function evaluations.
<i>convTol</i>	<i>float</i> The minimum change of the best objective value before the search terminates.
<i>stallLimit</i>	<i>integer</i> The maximum number of evaluations to search without an improvement.
<i>optConvTol</i>	<i>float</i> The maximum deviation from the best know fitness value before the search terminates.
<i>kwargs</i>	<i>ProblemParameters class arguments</i> Keyword arguments for the attributes of the ProblemParameters class. If not provided. The inherited 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

<i>self</i>	<i>GnoweeHeuristics pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
-------------	----------------------------------------------------------------------------------

#### 9.3.3.2 `def GnoweeHeuristics.GnoweeHeuristics.__str__( self )`

Human readable [GnoweeHeuristics](#) print function.

Parameters

<i>self</i>	<i>GnoweeHeuristics pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
-------------	----------------------------------------------------------------------------------

#### 9.3.3.3 `def GnoweeHeuristics.GnoweeHeuristics.comb_levy_flight( self, pop )`

Generate new children using truncated Levy flights permutation and inversion of current generation design parameters.

Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.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**

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.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**

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.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

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.

## 9.3.3.7 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."

Although logic originally designed for combinatorial variables, it works for all variables and is used for all here. The primary difference is the number of times that the crossover is performed.

## Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.

## 9.3.3.8 def GnoweeHeuristics.GnoweeHeuristics.initialize ( self, numSamples, sampleMethod )

Initialize the population according to the sampling method chosen.

## Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
-------------	---------------------------------------------------------------------------------

<i>numSamples</i>	<i>integer</i> The number of samples to be generated.
<i>sampleMethod</i>	<i>string</i> 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 <code>init_samples()</code> .

**Returns**

*list of arrays:* The initialized set of samples.

**9.3.3.9** `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 Optimization Algorithms"

**Parameters**

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.

**9.3.3.10** `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.

**Parameters**

<i>parents</i>	<i>list of parent objects</i> The current parents representing system designs.
<i>children</i>	<i>list of arrays</i> The children design variables representing new system designs.
<i>timeline</i>	<i>list of history objects</i> The histories of the optimization process containing best design, fitness, generation, and function evaluations.

<i>genUpdate</i>	<i>integer</i> Indicator for how many generations to increment the counter by. Genenerally 0 or 1.
<i>adoptedParents</i>	<i>list</i> 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.
<i>mhFrac</i>	<i>float</i> The Metropolis-Hastings fraction. A fraction of the otherwise discarded parents will be evaluated for acceptance against the greater population.
<i>randomParents</i>	<i>boolean</i> 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.

**9.3.3.11 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

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

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

where b is the size of the population.

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

Applies [simple\\_bounds\(\)](#) to ensure all solutions lie within the design space by adapting the step size to the size of the design space.

**Parameters**

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.

#### 9.3.3.12 `def GnoweeHeuristics.GnoweeHeuristics.three_opt ( self, pop )`

Generate new children using the `three_opt` operator.

Ideas adapted from: Lin and Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem"

##### Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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.

#### 9.3.3.13 `def GnoweeHeuristics.GnoweeHeuristics.two_opt ( self, pop )`

Generate new children using the `two_opt` operator.

Ideas adapted from: Lin and Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem"

##### Parameters

<i>self</i>	<i>GnoweeHeuristic pointer</i> The <a href="#">GnoweeHeuristics</a> pointer.
<i>pop</i>	<i>list of arrays</i> 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 generations to search without a decrease exceeding `convTol`.

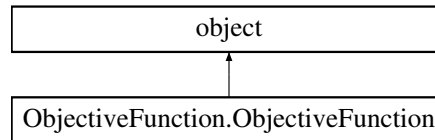
The documentation for this class was generated from the following file:

- `/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: Multitmodal,  $n$  dimensional.
- `def shifted\_ackley`  
Ackley Function: Multitmodal,  $n$  dimensional Ackley Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimum.
- `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 optimum.
- `def easom`  
Easom Function: Multimodal,  $n$ -dimensional.
- `def shifted\_easom`  
Easom Function: Multimodal,  $n$ -dimensional Easom Function that is shifted from the symmetric  $\pi$ ,  $\pi$  optimum.
- `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 optimum.*
- 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 optimum.*
- 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 optimum.*
- def [tsp](#)  
*Generic objective funtion to evaluate the TSP optimization by calculating total distance traveled.*

## Public Attributes

- [func](#)  
*function handle: The function handle for the objective function to be used for the optimization.*
- [objective](#)  
*integer, float, or numpy array: The desired outcome of the optimization.*

### 9.4.1 Detailed Description

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

<i>self</i>	<i>ObjectiveFunction</i> pointer The <a href="#">ObjectiveFunction</a> pointer.
<i>method</i>	<i>string</i> The name of the objective function to evaluate.
<i>objective</i>	<i>integer, float, or numpy array</i> The desired objective associated with the optimization. The chosen value and type must be compatible with the optiization 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

<i>self</i>	<i>ObjectiveFunction</i> pointer The <i>ObjectiveFunction</i> pointer.
-------------	---------------------------------------------------------------------------

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

Human readable *ObjectiveFunction* print function.

## Parameters

<i>self</i>	<i>ObjectiveFunction</i> pointer The <i>ObjectiveFunction</i> pointer.
-------------	---------------------------------------------------------------------------

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

Ackley Function: Multimodal, n dimensional.

Optimal example:

$u = [0, 0, 0, 0, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

## Parameters

<i>self</i>	<i>pointer</i> The <i>ObjectiveFunction</i> pointer.
<i>u</i>	<i>array</i> 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:

$u = [0, 0, 0, 0, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

## Parameters

<i>self</i>	<i>pointer</i> The <i>ObjectiveFunction</i> pointer.
<i>u</i>	<i>array</i> 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"

**Parameters**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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:

$u = [0, 0, 0, \dots, 0]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

**Parameters**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
-------------	------------------------------------------------------------------

<i>u</i>	<i>array</i>
----------	--------------

**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:

$u = [1, 1, 1, \dots, 1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

**Parameters**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i>

**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**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>funcName</i>	<i>string</i> A string identifying the objective function to be used.

**9.4.3.14 def ObjectiveFunction.ObjectiveFunction.shifted\_ackley ( self, u )**

Ackley Function: Multimodal, n dimensional Ackley Function that is shifted from the symmetric 0, 0, 0, ..., 0 optimum.

Optimal example:

$u = [0, 1, 2, 3, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"



## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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.

9.4.3.15 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 optimum.

Optimal example:

$u = [0, 1, 2, 3, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

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

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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  $\pi$ ,  $\pi$  optimum.

Optimal example:

$u = [\pi, \pi+1]$

fitness = 1.0

Taken from: "Nature-Inspired Optimization Algorithms"

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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 optimum.

Optimal example:

$u = [0, 1, 2, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

**Parameters**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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 optimum.

Optimal example:

$u = [0, 1, 2, \dots, n-1]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

**Parameters**

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i>

**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 optimum.

Optimal example:

$u = [1, 2, 3, \dots, n]$

fitness = 0.0

Taken from: "Nature-Inspired Optimization Algorithms"

#### Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i>

#### 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i>

## 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

<i>self</i>	<i>pointer</i> The <a href="#">ObjectiveFunction</a> pointer.
<i>u</i>	<i>array</i> 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.

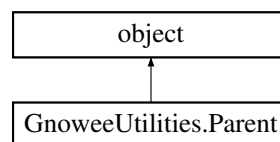
## 9.4.4.2 ObjectiveFunction.ObjectiveFunction.objective

*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.

## Parameters

<i>self</i>	<i>Parent</i> pointer The <i>Parent</i> pointer.
<i>variables</i>	<i>array</i> The set of variables representing a design solution.
<i>fitness</i>	<i>float</i> The assessed fitness for the current set of variables.
<i>changeCount</i>	<i>integer</i> The number of improvements to the current population member.
<i>stallCount</i>	<i>integer</i> 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

<i>self</i>	<i>Parent</i> pointer The <i>Parent</i> pointer.
-------------	-----------------------------------------------------

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

Human readable *Parent* print function.

## Parameters

<i>self</i>	<i>Parent</i> pointer The <i>Parent</i> 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.

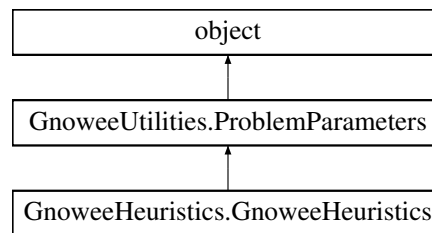
The documentation for this class was generated from the following file:

- /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.*
- `lb`  
*array: The lower bounds of the design variable(s).*
- `ub`  
*array: The upper bounds of the design variable(s).*
- `varType`  
*array: The type of variable for each position in the upper and lower bounds array.*



- [discreteVals](#)  
*Checks and cleans user inputs to be compatible with expectations from the [Gnowee](#) algorithm.*
- [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 integer variable truth array.*
- [dID](#)  
*array: The discrete variable truth array.*
- [xID](#)  
*array: The combinatorial 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

<i>self</i>	<a href="#">ProblemParameters</a> pointer The <a href="#">ProblemParameters</a> pointer.
<i>objective</i>	<i>ObjectiveFunction</i> object The optimization objective function to be used. Only a single objective function can be specified.

<i>constraints</i>	<i>list of Constraint objects</i> The constraints on the problem. Zero constraints can be specified as an empty list ( <code>[]</code> ), or multiple constraints can be specified as a list of Constraint objects.
<i>lowerBounds</i>	<i>array</i> 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 <code>varType</code> and <code>ub</code> .
<i>upperBounds</i>	<i>array</i> 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 <code>varType</code> and <code>lb</code> .
<i>varType</i>	<i>list or array</i> The type of variable for each position in the upper and lower bounds array. Discrete and combinatorial variables are to be included last as they are specified separately from the <code>lb/ub</code> through the <code>discreteVals</code> optional input. The order should be the same as shown below. Allowed values: 'c' = continuous over a given range (range specified in <code>lb</code> & <code>ub</code> ). 'i' = integer/binary (difference denoted by <code>ub/lb</code> ). 'f' = fixed design variable. Will not be considered of any permutation. 'd' = discrete where the allowed values are given by the option <code>discreteVals</code> <code>nxm</code> array with <code>n=#</code> of discrete variables and <code>m=#</code> of values that can be taken for each variable. 'x' = combinatorial. All of the variables denoted by <code>x</code> are assumed to be "swappable" in combinatorial permutations and assumed to take discrete values specified in by <code>discreteVals</code> . There must be at least two variables denoted as combinatorial. The algorithms are only set up to handle one set of combinatorial variables per optimization problem. Combinatorial variables should be specified last and as a contiguous group.
<i>discreteVals</i>	<i>list of list(s)</i> <code>nxm</code> with <code>n=#</code> of discrete and combinatorial variables and <code>m=#</code> 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 <code>discreteVals</code> could be specified as: <code>discreteVals = [[0.125, 0.25, 0.375], [0.25, 0.5, 0.75]]</code> For combinatorial problems, you must specify the same possible values that can be taken <code>n</code> times, where <code>n</code> is the number of different positions in the combinatorial sequence. suppose you had a gear that could be placed at position 2, 3, 4, or 5. The <code>discreteVals</code> would be specified as (assuming no other discretizes): <code>discreteVals = [[2, 3, 4, 5], [2, 3, 4, 5], [2, 3, 4, 5], [2, 3, 4, 5]]</code> \n <a href="#">Gnowee</a> will then map the optimization results to these allowed values.
<i>optimum</i>	<i>float</i> The global optimal solution.
<i>pltTitle</i>	<i>string</i> The title used for plotting the results of the optimization.
<i>histTitle</i>	<i>string</i> The plot title for the histogram of the optimization results.
<i>varNames</i>	<i>list of strings</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer.
-------------	------------------------------------------------------------------

## 9.6.3.2 def GnoweeUtilities.ProblemParameters.\_\_str\_\_( self )

Human readable [ProblemParameters](#) print function.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer.
-------------	------------------------------------------------------------------

## 9.6.3.3 def GnoweeUtilities.ProblemParameters.map\_from\_discretes ( self, variables )

Maps the discrete values to indices for sampling.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer. The <a href="#">Parent</a> pointer.
<i>variables</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer. The <a href="#">Parent</a> pointer.
<i>variables</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer.
-------------	------------------------------------------------------------------

9.6.3.6 `def GnoweeUtilities.ProblemParameters.set_preset_params ( self, funct, algorithm = "", dimension = 2 )`

Instantiates a [ProblemParameters](#) object and populates member variables from a set of predefined problem types.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer.
<i>funct</i>	<i>string</i> Name of function being optimized.
<i>algorithm</i>	<i>string</i> Name of optimization program used.
<i>dimension</i>	<i>integer</i> 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 discrete variable truth array.

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

9.6.4.4 `GnoweeUtilities.ProblemParameters.discreteVals`

Checks and cleans user inputs to be compatible with expectations from the [Gnowee](#) algorithm.

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

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">ProblemParameters</a> pointer.
-------------	------------------------------------------------------------------

#### 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 integer variable truth array.

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

#### 9.6.4.7 GnoweeUtilities.ProblemParameters.lb

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

#### 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 combinatorial variable truth array.

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

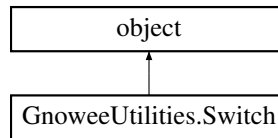
The documentation for this class was generated from the following file:

- /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

<i>self</i>	<i>pointer</i> The <a href="#">Switch</a> pointer.
-------------	-------------------------------------------------------

##### 9.7.2.2 `def GnoweeUtilities.Switch.match ( self, args )`

Indicate whether or not to enter a case suite.

## Parameters

<i>self</i>	<i>pointer</i> The <a href="#">Switch</a> pointer.
<i>*args</i>	<i>list</i> 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.

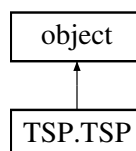
The documentation for this class was generated from the following file:

- /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.

Inheritance diagram for TSP.TSP:



## 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 coordinate 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

<i>self</i>	<a href="#">TSP</a> pointer The <a href="#">TSP</a> pointer.
<i>name</i>	<i>string</i> The name of the TSPLIB problem.
<i>dimension</i>	<i>integer</i> The number of nodes (cities) in the problem.
<i>nodes</i>	<i>list of lists</i> The coordinate pairs for each node.
<i>optimum</i>	<i>float</i> The optimal solution.

### 9.8.3 Member Function Documentation

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

[TSP](#) class param print function.

Parameters

<i>self</i>	<a href="#">TSP</a> pointer The <a href="#">TSP</a> pointer.
-------------	-----------------------------------------------------------------

9.8.3.2 `def TSP.TSP.__str__( self )`

Human readable [TSP](#) print function.



## Parameters

<i>self</i>	<a href="#">TSP</a> pointer The <a href="#">TSP</a> 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

<i>probParams</i>	<i>ProblemParameters</i> 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

<i>filename</i>	<i>string</i> 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 coordinate pairs for each node.

## 9.8.4.4 TSP.TSP.optimum

*float*: The optimal solution.

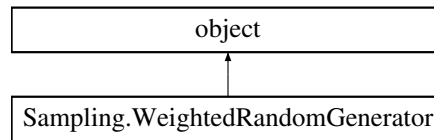
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Gnowee/src/[TSP.py](#)

## 9.9 Sampling.WeightedRandomGenerator Class Reference

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

Inheritance diagram for Sampling.WeightedRandomGenerator:



## 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 the 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

<i>self</i>	<i>pointer</i> The <a href="#">WeightedRandomGenerator</a> pointer.
<i>weights</i>	<i>array</i> 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

<i>self</i>	<i>pointer</i> The <a href="#">WeightedRandomGenerator</a> 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**

<i>self</i>	<i>pointer</i> The <a href="#">WeightedRandomGenerator</a> pointer.
-------------	------------------------------------------------------------------------

**Returns**

*integer*: The randomly selected index of the weights array.

**9.9.4 Member Data Documentation****9.9.4.1** `Sampling.WeightedRandomGenerator.totals`

*list or numpy array*: The ordinal ranking or data that is used to generate the 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.*

## Functions

- def [GnoweeHeuristics.simple\\_bounds](#)  
*Application of problem boundaries to generated solutions.*
- def [GnoweeHeuristics.rejection\\_bounds](#)  
*Application of problem boundaries to generated solutions.*
- def [GnoweeHeuristics.contains\\_sublist](#)  
*Find index of sublist, if it exists.*

## 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, multiplicative 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](#)

*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 permutations 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/-  
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](#)
- `__call__`
  - Sampling::WeightedRandomGenerator, [72](#)
- `__init__`
  - Constraints::Constraint, [30](#)
  - GnoweeHeuristics::GnoweeHeuristics, [40](#)
  - GnoweeUtilities, [17](#)
  - GnoweeUtilities::Event, [37](#)
  - GnoweeUtilities::Parent, [61](#)
  - GnoweeUtilities::ProblemParameters, [63](#)
  - ObjectiveFunction::ObjectiveFunction, [49](#)
  - Sampling::WeightedRandomGenerator, [72](#)
  - TSP::TSP, [70](#)
- `__iter__`
  - GnoweeUtilities::Switch, [68](#)
- `__repr__`
  - Constraints::Constraint, [30](#)
  - GnoweeHeuristics::GnoweeHeuristics, [41](#)
  - GnoweeUtilities::Event, [37](#)
  - GnoweeUtilities::Parent, [61](#)
  - GnoweeUtilities::ProblemParameters, [64](#)
  - ObjectiveFunction::ObjectiveFunction, [49](#)
  - TSP::TSP, [70](#)
- `__str__`
  - Constraints::Constraint, [31](#)
  - GnoweeHeuristics::GnoweeHeuristics, [41](#)
  - GnoweeUtilities::Event, [37](#)
  - GnoweeUtilities::Parent, [61](#)
  - GnoweeUtilities::ProblemParameters, [65](#)
  - ObjectiveFunction::ObjectiveFunction, [50](#)
  - TSP::TSP, [70](#)
- ackley
  - ObjectiveFunction::ObjectiveFunction, [50](#)
- alpha
  - GnoweeHeuristics::GnoweeHeuristics, [46](#)
- build\_prob\_params
  - TSP::TSP, [71](#)
- cID
  - GnoweeUtilities::ProblemParameters, [66](#)
- changeCount
  - GnoweeUtilities::Parent, [61](#)
- comb\_levy\_flight
  - GnoweeHeuristics::GnoweeHeuristics, [41](#)
- constraint
  - Constraints::Constraint, [36](#)
- Constraints, [13](#)
- constraints
  - GnoweeUtilities::ProblemParameters, [66](#)
- Constraints.Constraint, [29](#)
- Constraints::Constraint
  - `__init__`, [30](#)
  - `__repr__`, [30](#)
  - `__str__`, [31](#)
  - `constraint`, [36](#)
  - `func`, [36](#)
  - `get_penalty`, [31](#)
  - `greater_than`, [31](#)
  - `less_or_equal`, [31](#)
  - `less_than`, [32](#)
  - `mi_chemical_process`, [32](#)
  - `mi_pressure_vessel`, [32](#)
  - `mi_spring`, [33](#)
  - `pressure_vessel`, [33](#)
  - `set_constraint_func`, [34](#)
  - `speed_reducer`, [34](#)
  - `spring`, [35](#)
  - `welded_beam`, [35](#)
- cont\_levy\_flight
  - GnoweeHeuristics::GnoweeHeuristics, [42](#)
- contains\_sublist
  - GnoweeHeuristics, [15](#)
- convTol
  - GnoweeHeuristics::GnoweeHeuristics, [46](#)
- crossover
  - GnoweeHeuristics::GnoweeHeuristics, [42](#)
- dID
  - GnoweeUtilities::ProblemParameters, [66](#)
- dejong
  - ObjectiveFunction::ObjectiveFunction, [50](#)
- design

- GnoweeUtilities::Event, 37
- dimension
  - TSP::TSP, 71
- disc\_levy\_flight
  - GnoweeHeuristics::GnoweeHeuristics, 42
- discreteVals
  - GnoweeUtilities::ProblemParameters, 66
- easom
  - ObjectiveFunction::ObjectiveFunction, 51
- elite\_crossover
  - GnoweeHeuristics::GnoweeHeuristics, 43
- evaluations
  - GnoweeUtilities::Event, 37
- fall
  - GnoweeUtilities::Switch, 69
- fitness
  - GnoweeUtilities::Event, 37
  - GnoweeUtilities::Parent, 61
- fracDiscovered
  - GnoweeHeuristics::GnoweeHeuristics, 46
- fracElite
  - GnoweeHeuristics::GnoweeHeuristics, 46
- fracLevy
  - GnoweeHeuristics::GnoweeHeuristics, 46
- func
  - Constraints::Constraint, 36
  - ObjectiveFunction::ObjectiveFunction, 58
- gamma
  - GnoweeHeuristics::GnoweeHeuristics, 47
- generation
  - GnoweeUtilities::Event, 38
- get\_cdr\_permutations
  - Sampling, 22
- get\_penalty
  - Constraints::Constraint, 31
- Gnowee, 14, 27
  - main, 14
- GnoweeHeuristics, 15
  - contains\_sublist, 15
  - rejection\_bounds, 15
  - simple\_bounds, 16
- GnoweeHeuristics.GnoweeHeuristics, 38
- GnoweeHeuristics::GnoweeHeuristics
  - \_\_init\_\_, 40
  - \_\_repr\_\_, 41
  - \_\_str\_\_, 41
  - alpha, 46
  - comb\_levy\_flight, 41
  - cont\_levy\_flight, 42
  - convTol, 46
  - crossover, 42
  - disc\_levy\_flight, 42
  - elite\_crossover, 43
  - fracDiscovered, 46
  - fracElite, 46
  - fracLevy, 46
  - gamma, 47
  - initSampling, 47
  - initialize, 43
  - maxFevals, 47
  - maxGens, 47
  - mutate, 44
  - n, 47
  - optConvTol, 47
  - penalty, 47
  - population, 47
  - population\_update, 44
  - scalingFactor, 47
  - scatter\_search, 45
  - stallLimit, 47
  - three\_opt, 45
  - two\_opt, 46
- GnoweeUtilities, 17
  - \_\_init\_\_, 17
- GnoweeUtilities.Event, 36
- GnoweeUtilities.Parent, 59
- GnoweeUtilities.ProblemParameters, 62
- GnoweeUtilities.Switch, 68
- GnoweeUtilities::Event
  - \_\_init\_\_, 37
  - \_\_repr\_\_, 37
  - \_\_str\_\_, 37
  - design, 37
  - evaluations, 37
  - fitness, 37
  - generation, 38
- GnoweeUtilities::Parent
  - \_\_init\_\_, 60
  - \_\_repr\_\_, 61
  - \_\_str\_\_, 61
  - changeCount, 61
  - fitness, 61
  - stallCount, 61
  - variables, 61
- GnoweeUtilities::ProblemParameters
  - \_\_init\_\_, 63
  - \_\_repr\_\_, 64
  - \_\_str\_\_, 65
  - cID, 66
  - constraints, 66
  - dID, 66
  - discreteVals, 66
  - histTitle, 66
  - iID, 67
  - lb, 67
  - map\_from\_discretes, 65
  - map\_to\_discretes, 65
  - objective, 67
  - optimum, 67
  - pltTitle, 67
  - sanitize\_inputs, 65
  - set\_preset\_params, 66
  - ub, 67
  - varNames, 67

- varType, 67
- xID, 67
- GnoweeUtilities::Switch
  - \_\_iter\_\_, 68
  - fall, 69
  - match, 68
  - value, 69
- greater\_than
  - Constraints::Constraint, 31
- griewank
  - ObjectiveFunction::ObjectiveFunction, 51
- histTitle
  - GnoweeUtilities::ProblemParameters, 66
- iID
  - GnoweeUtilities::ProblemParameters, 67
- initSampling
  - GnoweeHeuristics::GnoweeHeuristics, 47
- initial\_samples
  - Sampling, 23
- initialize
  - GnoweeHeuristics::GnoweeHeuristics, 43
- lb
  - GnoweeUtilities::ProblemParameters, 67
- less\_or\_equal
  - Constraints::Constraint, 31
- less\_than
  - Constraints::Constraint, 32
- levy
  - Sampling, 23
- main
  - Gnowee, 14
- map\_from\_discretes
  - GnoweeUtilities::ProblemParameters, 65
- map\_to\_discretes
  - GnoweeUtilities::ProblemParameters, 65
- match
  - GnoweeUtilities::Switch, 68
- maxFevals
  - GnoweeHeuristics::GnoweeHeuristics, 47
- maxGens
  - GnoweeHeuristics::GnoweeHeuristics, 47
- mi\_chemical\_process
  - Constraints::Constraint, 32
  - ObjectiveFunction::ObjectiveFunction, 51
- mi\_pressure\_vessel
  - Constraints::Constraint, 32
  - ObjectiveFunction::ObjectiveFunction, 52
- mi\_spring
  - Constraints::Constraint, 33
  - ObjectiveFunction::ObjectiveFunction, 52
- mutate
  - GnoweeHeuristics::GnoweeHeuristics, 44
- n
  - GnoweeHeuristics::GnoweeHeuristics, 47
- NOLH
  - Sampling, 24
- name
  - TSP::TSP, 71
- next
  - Sampling::WeightedRandomGenerator, 73
- nodes
  - TSP::TSP, 71
- objective
  - GnoweeUtilities::ProblemParameters, 67
  - ObjectiveFunction::ObjectiveFunction, 59
- ObjectiveFunction, 18
  - prod, 18
- ObjectiveFunction.ObjectiveFunction, 48
- ObjectiveFunction::ObjectiveFunction
  - \_\_init\_\_, 49
  - \_\_repr\_\_, 49
  - \_\_str\_\_, 50
  - ackley, 50
  - dejong, 50
  - easom, 51
  - func, 58
  - griewank, 51
  - mi\_chemical\_process, 51
  - mi\_pressure\_vessel, 52
  - mi\_spring, 52
  - objective, 59
  - pressure\_vessel, 53
  - rastrigin, 53
  - rosenbrock, 54
  - set\_obj\_func, 54
  - shifted\_ackley, 54
  - shifted\_dejong, 55
  - shifted\_easom, 55
  - shifted\_griewank, 56
  - shifted\_rastrigin, 56
  - shifted\_rosenbrock, 56
  - speed\_reducer, 57
  - spring, 57
  - tsp, 58
  - welded\_beam, 58
- optConvTol
  - GnoweeHeuristics::GnoweeHeuristics, 47
- OptiPlot, 19
  - plot\_feval\_hist, 19
  - plot\_hist, 20
  - plot\_hist\_comp, 20
  - plot\_optimization, 20
  - plot\_tlf, 21
  - plot\_vars, 21
- optimum
  - GnoweeUtilities::ProblemParameters, 67
  - TSP::TSP, 71
- params
  - Sampling, 24
- penalty
  - GnoweeHeuristics::GnoweeHeuristics, 47

- plot\_feval\_hist
  - OptiPlot, 19
- plot\_hist
  - OptiPlot, 20
- plot\_hist\_comp
  - OptiPlot, 20
- plot\_optimization
  - OptiPlot, 20
- plot\_samples
  - Sampling, 24
- plot\_tlf
  - OptiPlot, 21
- plot\_vars
  - OptiPlot, 21
- pltTitle
  - GnoweeUtilities::ProblemParameters, 67
- population
  - GnoweeHeuristics::GnoweeHeuristics, 47
- population\_update
  - GnoweeHeuristics::GnoweeHeuristics, 44
- pressure\_vessel
  - Constraints::Constraint, 33
  - ObjectiveFunction::ObjectiveFunction, 53
- prod
  - ObjectiveFunction, 18
- rastrigin
  - ObjectiveFunction::ObjectiveFunction, 53
- read\_tsp
  - TSP::TSP, 71
- rejection\_bounds
  - GnoweeHeuristics, 15
- rosenbrock
  - ObjectiveFunction::ObjectiveFunction, 54
- Sampling, 22
  - get\_cdr\_permutations, 22
  - initial\_samples, 23
  - levy, 23
  - NOLH, 24
  - params, 24
  - plot\_samples, 24
  - tlf, 25
- Sampling.WeightedRandomGenerator, 71
- Sampling::WeightedRandomGenerator
  - \_\_call\_\_, 72
  - \_\_init\_\_, 72
  - next, 73
  - totals, 73
- sanitize\_inputs
  - GnoweeUtilities::ProblemParameters, 65
- scalingFactor
  - GnoweeHeuristics::GnoweeHeuristics, 47
- scatter\_search
  - GnoweeHeuristics::GnoweeHeuristics, 45
- set\_constraint\_func
  - Constraints::Constraint, 34
- set\_obj\_func
  - ObjectiveFunction::ObjectiveFunction, 54
- set\_preset\_params
  - GnoweeUtilities::ProblemParameters, 66
- shifted\_ackley
  - ObjectiveFunction::ObjectiveFunction, 54
- shifted\_dejong
  - ObjectiveFunction::ObjectiveFunction, 55
- shifted\_easom
  - ObjectiveFunction::ObjectiveFunction, 55
- shifted\_griewank
  - ObjectiveFunction::ObjectiveFunction, 56
- shifted\_rastrigin
  - ObjectiveFunction::ObjectiveFunction, 56
- shifted\_rosenbrock
  - ObjectiveFunction::ObjectiveFunction, 56
- simple\_bounds
  - GnoweeHeuristics, 16
- speed\_reducer
  - Constraints::Constraint, 34
  - ObjectiveFunction::ObjectiveFunction, 57
- spring
  - Constraints::Constraint, 35
  - ObjectiveFunction::ObjectiveFunction, 57
- stallCount
  - GnoweeUtilities::Parent, 61
- stallLimit
  - GnoweeHeuristics::GnoweeHeuristics, 47
- TSP, 26
- TSP.TSP, 69
- TSP::TSP
  - \_\_init\_\_, 70
  - \_\_repr\_\_, 70
  - \_\_str\_\_, 70
  - build\_prob\_params, 71
  - dimension, 71
  - name, 71
  - nodes, 71
  - optimum, 71
  - read\_tsp, 71
- three\_opt
  - GnoweeHeuristics::GnoweeHeuristics, 45
- tlf
  - Sampling, 25
- totals
  - Sampling::WeightedRandomGenerator, 73
- tsp
  - ObjectiveFunction::ObjectiveFunction, 58
- two\_opt
  - GnoweeHeuristics::GnoweeHeuristics, 46
- ub
  - GnoweeUtilities::ProblemParameters, 67
- value
  - GnoweeUtilities::Switch, 69
- varNames
  - GnoweeUtilities::ProblemParameters, 67
- varType
  - GnoweeUtilities::ProblemParameters, 67

## variables

GnoweeUtilities::Parent, [61](#)

## welded\_beam

Constraints::Constraint, [35](#)

ObjectiveFunction::ObjectiveFunction, [58](#)

## xID

GnoweeUtilities::ProblemParameters, [67](#)