
Opytimizer Documentation

Release 3.1.2

Gustavo de Rosa

Jul 19, 2024

PACKAGE REFERENCE

1	opoptimizer	3
2	opoptimizer.core	5
2.1	opoptimizer.core.agent	5
2.2	opoptimizer.core.block	6
2.3	opoptimizer.core.cell	7
2.4	opoptimizer.core.function	8
2.5	opoptimizer.core.node	9
2.6	opoptimizer.core.optimizer	11
2.7	opoptimizer.core.space	12
3	opoptimizer.functions	15
3.1	opoptimizer.functions.constrained	15
3.2	opoptimizer.functions.multi_objective	16
4	opoptimizer.math	19
4.1	opoptimizer.math.distribution	19
4.2	opoptimizer.math.general	20
4.3	opoptimizer.math.complex	21
4.4	opoptimizer.math.random	21
5	opoptimizer.optimizers	25
5.1	opoptimizer.optimizers.boolean	25
5.2	opoptimizer.optimizers.evolutionary	28
5.3	opoptimizer.optimizers.misc	44
5.4	opoptimizer.optimizers.population	49
5.5	opoptimizer.optimizers.science	65
5.6	opoptimizer.optimizers.social	83
5.7	opoptimizer.optimizers.swarm	90
6	opoptimizer.spaces	125
6.1	opoptimizer.spaces.boolean	125
6.2	opoptimizer.spaces.graph	125
6.3	opoptimizer.spaces.grid	125
6.4	opoptimizer.spaces.hyper_complex	126
6.5	opoptimizer.spaces.pareto	126
6.6	opoptimizer.spaces.search	127
6.7	opoptimizer.spaces.tree	127
7	opoptimizer.utils	131
7.1	opoptimizer.utils.callback	131

7.2	optimizer.utils.constant	133
7.3	optimizer.utils.exception	134
7.4	optimizer.utils.history	135
7.5	optimizer.utils.logging	135
8	optimizer.visualization	137
8.1	optimizer.visualization.convergence	137
8.2	optimizer.visualization.surface	137
9	Indices and tables	139
	Python Module Index	141
	Index	143

Did you ever reach a bottleneck in your computational experiments? Are you tired of selecting suitable parameters for a chosen technique? If yes, Opytimizer is the real deal! This package provides an easy-to-go implementation of meta-heuristic optimizations. From agents to a search space, from internal functions to external communication, we will foster all research related to optimizing stuff.

Use Opytimizer if you need a library or wish to:

- Create your own optimization algorithm;
- Design or use pre-loaded optimization tasks;
- Mix-and-match different strategies to solve your problem;
- Because it is fun to optimize things.

Opytimizer is compatible with: **Python 3.6+**.

OPYTIMIZER

```
class optimizer.Opytimizer(space: Space, optimizer: Optimizer, function: Function, save_agents: bool |  
                           None = False)
```

An Opytimizer class holds all the information needed in order to perform an optimization task.

```
__init__(space: Space, optimizer: Optimizer, function: Function, save_agents: bool | None = False) →  
None
```

Initialization method.

Parameters

- **space** – Space-child instance.
- **optimizer** – Optimizer-child instance.
- **function** – Function or Function-child instance.
- **save_agents** – Saves all agents in the search space.

```
property space: Space
```

Space-child instance (SearchSpace, HyperComplexSpace, etc).

```
property optimizer: Optimizer
```

Optimizer-child instance (PSO, BA, etc).

```
property function: Function
```

Function or Function-child instance (ConstrainedFunction, WeightedFunction, etc).

```
property history: History
```

Optimization history.

```
property iteration: int
```

Current iteration.

```
property total_iterations: int
```

Total number of iterations.

```
property evaluate_args: List[Any]
```

Converts the optimizer *evaluate* arguments into real variables.

Returns

List of real-attribute variables.

Return type

(List[Any])

property `update_args: List[Any]`

Converts the optimizer *update* arguments into real variables.

Returns

List of real-attribute variables.

Return type

(List[Any])

evaluate(*callbacks: List[Callback]*) → None

Wraps the *evaluate* pipeline with its corresponding callbacks.

Parameters

callbacks – List of callbacks.

update(*callbacks: List[Callback]*) → None

Wraps the *update* pipeline with its corresponding callbacks.

Parameters

callback – List of callbacks.

start(*n_iterations: int | None = 1, callbacks: List[Callback] | None = None*) → None

Starts the optimization task.

Args

n_iterations: Maximum number of iterations. *callback*: List of callbacks.

save(*file_path: str*) → None

Saves the optimization model to a dill (pickle) file.

Parameters

file_path – Path of file to be saved.

classmethod load(*file_path: str*) → None

Loads the optimization model from a dill (pickle) file without needing to instantiate the class.

Parameters

file_path – Path of file to be loaded.

OPYTIMIZER.CORE

Core is the core. Essentially, it is the parent of everything. You should find parent classes defining the basis of our structure. They should provide variables and methods that will help to construct other modules.

2.1 opytimizer.core.agent

Agent.

```
class opytimizer.core.agent.Agent(n_variables: int, n_dimensions: int, lower_bound: List[int | float],  
                                upper_bound: List[int | float], mapping: List[str] | None = None)
```

An Agent class for all optimization techniques.

```
__init__(n_variables: int, n_dimensions: int, lower_bound: List[int | float], upper_bound: List[int | float],  
         mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_variables** – Number of decision variables.
- **n_dimensions** – Number of dimensions.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **mapping** – String-based identifiers for mapping variables' names.

```
property n_variables: int
```

Number of decision variables.

```
property n_dimensions: int
```

Number of dimensions.

```
property position: ndarray
```

N-dimensional array of positions.

```
property fit: int | float
```

Fitness value.

Type

float

```
property lb: ndarray
```

Lower bounds.

property ub: ndarray

Upper bounds.

property ts: int

Timestamp of the agent.

property mapping: List[str]

Variables mapping.

property mapped_position: Dict[str, ndarray]

Dictionary mapping variables names and array of positions.

clip_by_bound() → None

Clips the agent's decision variables to the bounds limits.

fill_with_binary() → None

Fills the agent's decision variables with a binary distribution.

fill_with_static(values: ndarray) → None

Fills the agent's decision variables with static values. Note that this method ignore the agent's bounds, so use it carefully.

Parameters

values – Values to be filled.

fill_with_uniform() → None

Fills the agent's decision variables with a uniform distribution based on bounds limits.

2.2 opyimizer.core.block

Block.

class opyimizer.core.block.**Block**(type: str, pointer: callable, n_input: int, n_output: int)

A Block serves as the foundation class for all graph-based optimization techniques.

__init__(type: str, pointer: callable, n_input: int, n_output: int) → None

Initialization method.

Parameters

- **type** – Type of the block.
- **pointer** – Any type of callable to be applied when block is called.
- **n_input** – Number of input arguments.
- **n_output** – Number of output arguments.

__call__(*args) → callable

Callable to avoid using the *pointer* property.

Returns

Input arguments applied to callable *pointer*.

property type: str

Type of the block.

property pointer: callable

Points to the actual function when block is called.

property n_input: int

Number of input arguments.

property n_output: int

Number of output arguments.

class opyimizer.core.block.**InputBlock**(*n_input: int, n_output: int*)

An InputBlock defines a block that is only used for entry points.

__init__(*n_input: int, n_output: int*) → None

Initialization method.

Parameters

- **n_input** – Number of input arguments.
- **n_output** – Number of output arguments.

class opyimizer.core.block.**InnerBlock**(*pointer: callable, n_input: int, n_output: int*)

An InnerBlock defines a block that is used for inner points (between input and output).

__init__(*pointer: callable, n_input: int, n_output: int*) → None

Initialization method.

Parameters

- **pointer** – Any type of callable to be applied when block is called.
- **n_input** – Number of input arguments.
- **n_output** – Number of output arguments.

class opyimizer.core.block.**OutputBlock**(*n_input: int, n_output: int*)

An OutputBlock defines a block that is only used for output points.

__init__(*n_input: int, n_output: int*) → None

Initialization method.

Parameters

- **n_input** – Number of input arguments.
- **n_output** – Number of output arguments.

2.3 opyimizer.core.cell

Cell.

class opyimizer.core.cell.**Cell**(*blocks: Block, edges: Tuple[Block, Block]*)

A Cell serves a Direct Acyclic Graph (DAG) which holds blocks as nodes and edges that connects operation paths between the nodes.

__init__(*blocks: Block, edges: Tuple[Block, Block]*) → None

Initialization method.

Parameters

- **type** – Type of the block.
- **pointer** – Any type of callable to be applied when block is called.

__call__(*args) → Generator
Performs a forward pass over the cell.

Returns
Output for each possible path in DAG.

Return type
(Generator)

property input_idx: int
Index of the input node.

property output_idx: int
Index of the output node.

property valid: bool
Whether cell is valid or not.

2.4 opyoptimizer.core.function

Single-objective functions.

class opyoptimizer.core.function.**Function**(pointer: callable)

A Function class used to hold single-objective functions.

__init__(pointer: callable) → None
Initialization method.

Parameters

pointer – Pointer to a function that will return the fitness value.

__call__(x: ndarray) → float
Callable to avoid using the *pointer* property.

Parameters

x – Array of positions.

Returns

Single-objective function fitness.

Return type

(float)

property pointer: callable
Points to the actual function.

Type

callable

property name: str
Name of the function.

property built: bool
Indicates whether the function is built.

2.5 opyimizer.core.node

Node.

```
class opyimizer.core.node.Node(name: str | int, category: str, value: ndarray | None = None, left: Node | None = None, right: Node | None = None, parent: Node | None = None)
```

A Node instance is used for composing tree-based structures.

```
__init__(name: str | int, category: str, value: ndarray | None = None, left: Node | None = None, right: Node | None = None, parent: Node | None = None) → None
```

Initialization method.

Parameters

- **name** – Name of the node (e.g., it should be the terminal identifier or function name).
- **category** – Category of the node (e.g., `TERMINAL` or `FUNCTION`).
- **value** – Value of the node (only used if it is a terminal).
- **left** – Pointer to node's left child.
- **right** – Pointer to node's right child.
- **parent** – Pointer to node's parent.

```
__repr__() → str
```

Representation of a formal string.

```
__str__() → str
```

Representation of an informal string.

```
property name: str | int
```

Name of the node.

```
property category: str
```

Category of the node.

```
property value: ndarray
```

Value of the node.

Type

np.array

```
property left: Node
```

Pointer to the node's left child.

```
property right: Node
```

Pointer to the node's right child.

```
property parent: Node
```

Pointer to the node's parent.

```
property flag: bool
```

Flag to identify whether the node is a left child.

```
property min_depth: int
```

Minimum depth of node.

```
property max_depth: int
```

Maximum depth of node.

property n_leaves: int

Number of leaves node.

property n_nodes: int

Number of nodes.

property position: ndarray

Position after traversing the node.

property post_order: List[Node]

Traverses the node in post-order.

property pre_order: List[Node]

Traverses the node in pre-order.

find_node(position: int) → Node

Finds a node at a given position.

Parameters

position – Position of the node.

Returns

Node at desired position.

Return type

(Node)

`opytimizer.core.node._build_string(node: Node) → str`

Builds a formatted string for displaying the nodes.

References

https://github.com/joowani/binarytree/blob/master/binarytree/__init__.py#L153

Parameters

node – An instance of the Node class (can be a tree of Nodes).

Returns

Formatted string ready to be printed.

Return type

(str)

`opytimizer.core.node._evaluate(node: Node) → ndarray`

Evaluates a node and outputs its solution array.

Parameters

node – An instance of the Node class (can be a tree of Nodes).

Returns

Output solution of size (n_variables x n_dimensions).

Return type

(np.ndarray)

`opytimizer.core.node._properties(node: Node) → Dict[str, Any]`

Traverses the node and returns some useful properties.

Parameters

node – An instance of the Node class (can be a tree of Nodes).

Returns

Dictionary containing some useful properties: *min_depth*, *max_depth*, *n_leaves* and *n_nodes*.

Return type

(Dict[str, Any])

2.6 opytimizer.core.optimizer

Optimizer.

class opytimizer.core.optimizer.Optimizer

An Optimizer class that holds meta-heuristics-related properties and methods.

__init__() → None

Initialization method.

property algorithm: str

Algorithm's name.

Type

str

property built: bool

Indicates whether the optimizer is built.

property params: Dict[str, Any]

Key-value parameters.

build(*params: Dict[str, Any]*) → None

Builds the object by creating its parameters.

Parameters

params – Key-value parameters to the meta-heuristic.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

This method is called before the optimization procedure and makes sure that the additional variable is available as a property.

evaluate(*space: Space, function: Function*) → None

Evaluates the search space according to the objective function.

If you need a specific evaluate method, please re-implement it on child's class.

Also, note that function only accept arguments that are found on Opytizer class.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object serving as an objective function.

update() → None

Updates the agents' position array.

As each child has a different procedure of update, you will need to implement it directly on its class.

Also, note that function only accept arguments that are found on Opytizer class.

2.7 opyoptimizer.core.space

Search space.

```
class opyoptimizer.core.space.Space(n_agents: int | None = 1, n_variables: int | None = 1, n_dimensions: int  
                                   | None = 1, lower_bound: float | List | Tuple | ndarray | None = 0.0,  
                                   upper_bound: float | List | Tuple | ndarray | None = 1.0, mapping:  
                                   List[str] | None = None)
```

A Space class for agents, variables and methods related to the search space.

```
__init__(n_agents: int | None = 1, n_variables: int | None = 1, n_dimensions: int | None = 1, lower_bound:  
         float | List | Tuple | ndarray | None = 0.0, upper_bound: float | List | Tuple | ndarray | None = 1.0,  
         mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_agents** – Number of agents.
- **n_variables** – Number of decision variables.
- **n_dimensions** – Dimension of search space.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **mapping** – String-based identifiers for mapping variables' names.

property n_agents: int

Number of agents.

property n_variables: int

Number of decision variables.

property n_dimensions: int

Number of search space dimensions.

property lb: ndarray

Minimum possible values.

property ub: ndarray

Maximum possible values.

property mapping: List[str]

Variables mapping.

property agents: List[Agent]

Agents that belongs to the space.

Type

list

property best_agent: Agent

Best agent.

Type

Agent

property built: bool

Indicates whether the space is built.

_create_agents() → None

Creates a list of agents.

_initialize_agents() → None

Initializes agents with their positions and defines a best agent.

As each child has a different procedure of initialization, you will need to implement it directly on its class.

build() → None

Builds the object by creating and initializing the agents.

clip_by_bound() → None

Clips the agents' decision variables to the bounds limits.

Core package for all common opyimizer modules.

OPYTIMIZER.FUNCTIONS

Instead of using raw and straightforward functions, why not try this module? Compose high-level abstract functions or even new function-based ideas in order to solve your problems. Note that for now, we will only support multi-objective function strategies.

3.1 opytimizer.functions.constrained

Constrained single-objective functions.

```
class opytimizer.functions.constrained.ConstrainedFunction(pointer: List[callable], constraints:
                                                         List[callable], penalty: float | None =
                                                         0.0)
```

A ConstrainedFunction class used to hold constrained single-objective functions.

```
__init__(pointer: List[callable], constraints: List[callable], penalty: float | None = 0.0) → None
```

Initialization method.

Parameters

- **pointer** – Pointer to a function that will return the fitness value.
- **constraints** – Constraints to be applied to the fitness function.
- **penalty** – Penalization factor when a constraint is not valid.

```
property constraints: List[callable]
```

Constraints to be applied to the fitness function.

```
property penalty: float
```

Penalization factor.

```
__call__(x: ndarray) → float
```

Callable to avoid using the *pointer* property.

Parameters

x – Array of positions.

Returns

Constrained single-objective function fitness.

Return type

(float)

3.2 opyimizer.functions.multi_objective

3.2.1 opyimizer.functions.multi_objective.standard

Standard multi-objective functions.

```
class opyimizer.functions.multi_objective.standard.MultiObjectiveFunction(functions:  
                                                                           List[callable])
```

A MultiObjectiveFunction class used to hold multi-objective functions.

```
__init__(functions: List[callable]) → None
```

Initialization method.

Parameters

functions – Pointers to functions that will return the fitness value.

```
__call__(x: ndarray) → float
```

Callable to avoid using the *pointer* property.

Parameters

x – Array of positions.

Returns

Multi-objective function fitness.

Return type

(float)

```
property functions: List[callable]
```

Function's instances.

3.2.2 opyimizer.functions.multi_objective.weighted

Multi-objective weighted functions.

```
class opyimizer.functions.multi_objective.weighted.MultiObjectiveWeightedFunction(functions:  
                                                                           List[callable],  
                                                                           weights:  
                                                                           List[float])
```

A MultiObjectiveWeightedFunction class used to hold multi-objective weighted functions.

```
__init__(functions: List[callable], weights: List[float]) → None
```

Initialization method.

Parameters

- **functions** – Pointers to functions that will return the fitness value.

- **weights** – Weights for weighted-sum strategy.

```
__call__(x: ndarray) → float
```

Callable to avoid using the *pointer* property.

Parameters

x – Array of positions.

Returns

Multi-objective weighted function fitness.

Return type

(float)

property weights: `List[float]`

Functions' weights.

Multi-objective functions package for all common opyimizer modules. Functions package for all common opyimizer modules.

OPYTIMIZER.MATH

Just because we are computing stuff does not mean that we do not need math. Math is the mathematical package containing low-level math implementations. From random numbers to distribution generation, you can find your needs on this module.

4.1 opytimizer.math.distribution

Distribution-based mathematical generators.

`opytimizer.math.distribution.generate_bernoulli_distribution`(*prob: float | None = 0.0, size: int | None = 1*) → ndarray

Generates a Bernoulli distribution based on an input probability.

Parameters

- **prob** – Probability of distribution.
- **size** – Size of array.

Returns

Bernoulli distribution n-dimensional array.

Return type

(np.ndarray)

`opytimizer.math.distribution.generate_choice_distribution`(*n: int | None = 1, probs: ndarray | None = None, size: int | None = 1*) → ndarray

Generates a random choice distribution based on probabilities.

Parameters

- **n** – Amount of values to be picked from.
- **probs** – Array of probabilities.
- **size** – Size of array.

Returns

Choice distribution array.

Return type

(np.ndarray)

`opytimizer.math.distribution.generate_levy_distribution`(*beta: float | None = 0.1, size: int | None = 1*) → ndarray

Generates a n-dimensional array based on a Lévy distribution.

References

X.-S. Yang and S. Deb. Computers & Operations Research. Multiobjective Cuckoo Search for Design Optimization (2013).

Parameters

- **beta** – Skewness parameter.
- **size** – Size of array.

Returns

Lévy distribution n-dimensional array.

Return type

(np.ndarray)

4.2 opyoptimizer.math.general

General-based mathematical functions.

`opyoptimizer.math.general.euclidean_distance(x: ndarray, y: ndarray) → float`

Calculates the Euclidean distance between two n-dimensional points.

Parameters

- **x** – N-dimensional point.
- **y** – N-dimensional point.

Returns

Euclidean distance between x and y.

Return type

(float)

`opyoptimizer.math.general.kmeans(x: ndarray, n_clusters: int | None = 1, max_iterations: int | None = 100, tol: float | None = 0.0001) → ndarray`

Performs the K-Means clustering over the input data.

Parameters

- **x** – Input array with a shape equal to (n_samples, n_variables, n_dimensions).
- **n_clusters** – Number of clusters.
- **max_iterations** – Maximum number of clustering iterations.
- **tol** – Tolerance value to stop the clustering.

Returns

An array holding the assigned cluster per input sample.

Return type

(np.ndarray)

`opyoptimizer.math.general.n_wise(x: List[Any], size: int | None = 2) → Iterable`

Iterates over an iterator and returns n-wise samples from it.

Parameters

- **x (list)** – Values to be iterated over.
- **size** – Amount of samples per iteration.

Returns

N-wise samples from the iterator.

Return type

(Iterable)

`opyoptimizer.math.general.tournament_selection(fitness: List[float], n: int, size: int | None = 2) → array`

Selects n-individuals based on a tournament selection.

Parameters

- **fitness** (*list*) – List of individuals fitness.
- **n** – Number of individuals to be selected.
- **size** – Tournament size.

Returns

Indexes of selected individuals.

Return type

(np.array)

`opyoptimizer.math.general.weighted_wheel_selection(weights: List[float]) → int`

Selects an individual from a weight-based roulette.

Parameters

weights – List of individuals weights.

Returns

Weight-based roulette individual.

Return type

(int)

4.3 opyoptimizer.math.complex

4.4 opyoptimizer.math.random

Random-based mathematical generators.

`opyoptimizer.math.random.generate_binary_random_number(size: int | None = 1) → ndarray`

Generates a binary random number or array based on an uniform distribution.

Parameters

size – Size of array.

Returns

A binary random number or array.

Return type

(np.ndarray)

`opyoptimizer.math.random.generate_exponential_random_number(scale: float | None = 1.0, size: int | None = 1) → ndarray`

Generates a random number or array based on an exponential distribution.

Parameters

- **scale** – Scaling of the distribution.
- **size** – Size of array.

Returns

An exponential random number or array.

Return type

(np.ndarray)

`opytimizer.math.random.generate_gamma_random_number(shape: float | None = 1.0, scale: float | None = 1.0, size: int | None = 1) → ndarray`

Generates an Erlang distribution based on gamma values.

Parameters

- **shape** – Shape parameter.
- **scale** – Scaling of the distribution.
- **size** – Size of array.

Returns

An Erlang distribution array.

Return type

(np.ndarray)

`opytimizer.math.random.generate_integer_random_number(low: int | None = 0, high: int | None = 1, exclude_value: int | None = None, size: int | None = None) → ndarray`

Generates a random number or array based on an integer distribution.

Parameters

- **low** – Lower interval.
- **high** – Higher interval.
- **exclude_value** – Value to be excluded from array.
- **size** – Size of array.

Returns

An integer random number or array.

Return type

(np.ndarray)

`opytimizer.math.random.generate_uniform_random_number(low: float | None = 0.0, high: float | None = 1.0, size: int | None = 1) → ndarray`

Generates a random number or array based on a uniform distribution.

Parameters

- **low** – Lower interval.
- **high** – Higher interval.
- **size** – Size of array.

Returns

A uniform random number or array.

Return type

(np.ndarray)

`opytimizer.math.random.generate_gaussian_random_number`(*mean: float | None = 0.0, variance: float | None = 1.0, size: int | None = 1*) → ndarray

Generates a random number or array based on a gaussian distribution.

Parameters

- **mean** – Gaussian’s mean value.
- **variance** – Gaussian’s variance value.
- **size** – Size of array.

Returns

A gaussian random number or array.

Return type

(np.ndarray)

Mathematical package for all common opytimizer modules.

OPYTIMIZER.OPTIMIZERS

This is why we are called Opytimizer. This is the heart of heuristics, where you can find a large number of meta-heuristics, optimization techniques, anything that can be called an optimizer. Please take a look at the [available optimizers](<https://github.com/gugarosa/opytimizer/wiki/Types-of-Optimizers>).

5.1 opytimizer.optimizers.boolean

5.1.1 opytimizer.optimizers.boolean.bmrfo

Boolean Manta Ray Foraging Optimization.

class opytimizer.optimizers.boolean.bmrfo.**BMRFO**(*params: Dict[str, Any] | None = None*)

A BMRFO class, inherited from Optimizer.

This is the designed class to define boolean MRFO-related variables and methods.

References

Publication pending.

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property S: ndarray

Somersault foraging.

_cyclone_foraging(*agents: List[Agent], best_position: ndarray, i: int, iteration: int, n_iterations: int*) → ndarray

Performs the cyclone foraging procedure.

Parameters

- **agents** – List of agents.
- **best_position** – Global best position.
- **i** – Current agent's index.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

A new cyclone foraging.

Return type

(np.ndarray)

_chain_foraging(*agents*: List[Agent], *best_position*: ndarray, *i*: int) → ndarray

Performs the chain foraging procedure.

Parameters

- **agents** – List of agents.
- **best_position** – Global best position.
- **i** – Current agent's index.

Returns

A new chain foraging.

Return type

(np.ndarray)

_somersault_foraging(*position*: ndarray, *best_position*: ndarray) → ndarray

Performs the somersault foraging procedure.

Parameters

- **position** – Agent's current position.
- **best_position** – Global best position.

Returns

A new somersault foraging.

Return type

(np.ndarray)

update(*space*: Space, *function*: Function, *iteration*: int, *n_iterations*: int) → None

Wraps Boolean Manta Ray Foraging Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.1.2 opytimizer.optimizers.boolean.bps

Boolean Particle Swarm Optimization.

class opytimizer.optimizers.boolean.bps.BPSO(*params*: Dict[str, Any] | None = None)

A BPSO class, inherited from Optimizer.

This is the designed class to define boolean PSO-related variables and methods.

References

F. Afshinmanesh, A. Marandi and A. Rahimi-Kian. A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System. IEEE International Conference on Smart Technologies (2005).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c1: ndarray

Cognitive constant.

property c2: ndarray

Social constant.

property local_position: ndarray

Array of local positions.

property velocity: ndarray

Array of velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

evaluate(*space: Space, function: Function*) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(*space: Space*) → None

Wraps Boolean Particle Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.1.3 opyoptimizer.optimizers.boolean.umd

Univariate Marginal Distribution Algorithm.

class opyoptimizer.optimizers.boolean.umd.UMDA(*params: Dict[str, Any] | None = None*)

An UMDA class, inherited from Optimizer.

This is the designed class to define UMDA-related variables and methods.

References

H. Mühlenbein. The equation for response to selection and its use for prediction. Evolutionary Computation (1997).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property p_selection: float

Probability of selection.

property lower_bound: float

Distribution lower bound.

property upper_bound: float

Distribution upper bound.

_calculate_probability(agents: List[Agent]) → ndarray

Calculates probabilities based on pre-selected agents' variables occurrence (eq. 47).

Parameters

agents – List of pre-selected agents.

Returns

Probability of variables occurrence.

Return type

(np.ndarray)

_sample_position(probs: ndarray) → ndarray

Samples new positions according to their probability of occurrence (eq. 53).

Parameters

probs – Array of probabilities.

Returns

New sampled position.

Return type

(np.ndarray)

update(space: Space) → None

Wraps Univariate Marginal Distribution Algorithm over all agents and variables.

Parameters

space – Space containing agents and update-related information.

A boolean package for all common opyimizer modules. It contains implementations of boolean-based optimizers.

5.2 opyimizer.optimizers.evolutionary

5.2.1 opyimizer.optimizers.evolutionary.bsa

Backtracking Search Optimization Algorithm.

class opyimizer.optimizers.evolutionary.bsa.BSA(params: Dict[str, Any] | None = None)

A BSA class, inherited from Optimizer.

This is the designed class to define BSOA-related variables and methods.

References

P. Civicioglu. Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation (2013).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property F: float

Experience from previous generation.

property mix_rate: int

Number of non-crosses.

property old_agents: List[Agent]

List of historical agents.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_permute(*agents: List[Agent]*) → None

Performs the permuting operator.

Parameters

agents – List of agents.

_mutate(*agents: List[Agent]*) → List[Agent]

Performs the mutation operator.

Parameters

agents – List of agents.

Returns

A list holding the trial agents.

Return type

(List[Agent])

_crossover(*agents: List[Agent], trial_agents: List[Agent]*) → None

Performs the crossover operator.

Parameters

- **agents** – List of agents.
- **trial_agents** – List of trial agents.

update(*space: Space, function: Function*) → None

Wraps Backtracking Search Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.2 opytimizer.optimizers.evolutionary.de

Differential Evolution.

class `optimizer.optimizers.evolutionary.de.DE`(*params: Dict[str, Any] | None = None*)

A DE class, inherited from `Optimizer`.

This is the designed class to define DE-related variables and methods.

References

R. Storn. On the usage of differential evolution for function optimization. Proceedings of North American Fuzzy Information Processing (1996).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property CR: float

Crossover probability.

property F: float

Differential weight.

_mutate_agent(*agent: Agent, alpha: Agent, beta: Agent, gamma: Agent*) → *Agent*

Mutates a new agent based on pre-picked distinct agents (eq. 4).

Parameters

- **agent** – Current agent.
- **alpha** – 1st picked agent.
- **beta** – 2nd picked agent.
- **gamma** – 3rd picked agent.

Returns

A mutated agent.

Return type

(*Agent*)

update(*space: Space, function: Function*) → None

Wraps Differential Evolution over all agents and variables (eq. 1-4).

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.3 optimizer.optimizers.evolutionary.ep

Evolutionary Programming.

class `optimizer.optimizers.evolutionary.ep.EP`(*params: Dict[str, Any] | None = None*)

An EP class, inherited from `Optimizer`.

This is the designed class to define EP-related variables and methods.

References

A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Natural Computing Series (2013).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property bout_size: float

Size of bout during the tournament selection.

property clip_ratio: float

Clipping ratio to helps the algorithm's convergence.

property strategy: ndarray

Array of strategies.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_mutate_parent(*agent: Agent, index: int, function: Function*) → *Agent*

Mutates a parent into a new child (eq. 5.1).

Parameters

- **agent** – An agent instance to be reproduced.
- **index** – Index of current agent.
- **function** – A Function object that will be used as the objective function.

Returns

A mutated child.

Return type

(*Agent*)

_update_strategy(*index: int, lower_bound: ndarray, upper_bound: ndarray*) → ndarray

Updates the strategy and performs a clipping process to help its convergence (eq. 5.2).

Parameters

- **index** – Index of current agent.
- **lower_bound** – An array holding the lower bounds.
- **upper_bound** – An array holding the upper bounds.

Returns

The updated strategy.

Return type

(np.ndarray)

update(*space: Space, function: Function*) → None

Wraps Evolutionary Programming over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.

- **function** – A Function object that will be used as the objective function.

5.2.4 optimizer.optimizers.evolutionary.es

Evolution Strategies.

class optimizer.optimizers.evolutionary.es.**ES**(*params: Dict[str, Any] | None = None*)

An ES class, inherited from Optimizer.

This is the designed class to define ES-related variables and methods.

References

T. Bäck and H.-P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation (1993).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property child_ratio: float

Ratio of children in the population.

property n_children: int

Number of children.

property strategy: ndarray

Array of strategies.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_mutate_parent(*agent: Agent, index: int, function: Function*) → *Agent*

Mutates a parent into a new child (eq. 2).

Parameters

- **agent** – An agent instance to be reproduced.
- **index** – Index of current agent.
- **function** – A Function object that will be used as the objective function.

Returns

A mutated child.

Return type

(*Agent*)

_update_strategy(*index: int*) → ndarray

Updates the strategy (eq. 5-10).

Parameters

index – Index of current agent.

Returns

The updated strategy.

Return type
(np.ndarray)

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Evolution Strategies over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.5 opyoptimizer.optimizers.evolutionary.foa

Forest Optimization Algorithm.

class opyoptimizer.optimizers.evolutionary.foa.FOA(*params*: *Dict[str, Any] | None = None*)

A FOA class, inherited from Optimizer.

This is the designed class to define FOA-related variables and methods.

References

M. Ghaemi, Mohammad-Reza F.-D. Forest Optimization Algorithm. Expert Systems with Applications (2014).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property life_time: int

Maximum age of trees.

property area_limit: int

Maximum number of trees in the florest.

property LSC: int

Local Seeding Changes.

property GSC: int

Global Seeding Changes.

property transfer_rate: float

Global seeding percentage.

property age: List[int]

Trees ages.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_local_seeding(*space*: [Space](#), *function*: [Function](#)) → None

Performs the local seeding on zero-aged trees.

Parameters

- **space** – A Space object containing meta-information.
- **function** – A Function object that will be used as the objective function.

_population_limiting(*space*: Space) → List[Agent]

Limits the population by removing old trees.

Parameters

space – A Space object containing meta-information.

Returns

A list of candidate trees that were removed from the forest.

Return type

(List[Agent])

_global_seeding(*space*: Space, *function*: Function, *candidate*: List[Agent]) → None

Performs the global seeding.

Parameters

- **space** – A Space object containing meta-information.
- **function** – A Function object that will be used as the objective function.
- **candidate** – Candidate trees.

update(*space*: Space, *function*: Function) → None

Wraps Forest Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.6 opytimizer.optimizers.evolutionary.ga

Genetic Algorithm.

class opytimizer.optimizers.evolutionary.ga.GA(*params*: Dict[str, Any] | None = None)

An GA class, inherited from Optimizer.

This is the designed class to define GA-related variables and methods.

References

M. Mitchell. An introduction to genetic algorithms. MIT Press (1998).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property p_selection: float

Probability of selection.

property p_mutation: float

Probability of mutation.

property p_crossover: float

Probability of crossover.

_roulette_selection(*n_agents*: int, *fitness*: List[float]) → List[int]

Performs a roulette selection on the population (p. 8).

Parameters

- **n_agents** – Number of agents allowed in the space.
- **fitness** – A fitness list of every agent.

Returns

The selected indexes of the population.

Return type

(List[int])

_crossover(*father*: Agent, *mother*: Agent) → Tuple[Agent, Agent]

Performs the crossover between a pair of parents (p. 8).

Parameters

- **father** – Father to produce the offsprings.
- **mother** – Mother to produce the offsprings.

Returns

Two generated offsprings based on parents.

Return type

(Tuple[Agent, Agent])

_mutation(*alpha*: Agent, *beta*: Agent) → Tuple[Agent, Agent]

Performs the mutation over offsprings (p. 8).

Parameters

- **alpha** – First offspring.
- **beta** – Second offspring.

Returns

Two mutated offsprings.

Return type

(Tuple[Agent, Agent])

update(*space*: Space, *function*: Function) → None

Wraps Genetic Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.7 opyimizer.optimizers.evolutionary.gp

Genetic Programming.

class opyimizer.optimizers.evolutionary.gp.GP(*params*: Dict[str, Any] | None = None)

A GP class, inherited from Optimizer.

This is the designed class to define GP-related variables and methods.

References

J. Koza. Genetic programming: On the programming of computers by means of natural selection (1992).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property p_reproduction: float

Probability of reproduction.

property p_mutation: float

Probability of mutation.

property p_crossover: float

Probability of crossover.

property pruning_ratio: float

Nodes' pruning ratio.

_prune_nodes(*n_nodes: int*) → int

Prunes the amount of possible nodes used for mutation and crossover.

Parameters

n_nodes – Number of current nodes.

Returns

Amount of pruned nodes.

Return type

(int)

_reproduction(*space: TreeSpace*) → None

Reproduces a number of individuals pre-selected through a tournament procedure (p. 99).

Parameters

space – A TreeSpace object.

_mutation(*space: TreeSpace*) → None

Mutates a number of individuals pre-selected through a tournament procedure.

Parameters

space – A TreeSpace object.

_mutate(*space: TreeSpace, tree: Node, max_nodes: int*) → *Node*

Actually performs the mutation on a single tree (p. 105).

Parameters

- **space** – A TreeSpace object.
- **trees** – A Node instance to be mutated.
- **max_nodes** – Maximum number of nodes to be searched.

Returns

A mutated tree.

Return type

(*Node*)

_crossover(*space*: [TreeSpace](#)) → None

Crossover a number of individuals pre-selected through a tournament procedure (p. 101).

Parameters

space – A [TreeSpace](#) object.

_cross(*father*: [Node](#), *mother*: [Node](#), *max_father*: int, *max_mother*: int) → [Tuple](#)[[Node](#), [Node](#)]

Actually performs the crossover over a father and mother nodes.

Parameters

- **father** – A father’s node to be crossed.
- **mother** – A mother’s node to be crossed.
- **max_father** – Maximum of nodes from father to be used.
- **max_mother** – Maximum of nodes from mother to be used.

Returns

Two offsprings based on the crossover operator.

Return type

([Tuple](#)[[Node](#), [Node](#)])

evaluate(*space*: [Space](#), *function*: [Function](#)) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A [TreeSpace](#) object.
- **function** – A [Function](#) object that will be used as the objective function.

update(*space*: [Space](#)) → None

Wraps Genetic Programming over all trees and variables.

Parameters

space – [TreeSpace](#) containing agents and update-related information.

5.2.8 opyoptimizer.optimizers.evolutionary.hs

Harmony Search-based algorithms.

class `opyoptimizer.optimizers.evolutionary.hs.HS`(*params*: [Dict](#)[*str*, *Any*] | None = None)

A HS class, inherited from [Optimizer](#).

This is the designed class to define HS-related variables and methods.

References

Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. Simulation (2001).

__init__(*params*: [Dict](#)[*str*, *Any*] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property **HMCR**: float

Harmony memory considering rate.

property PAR: float

Pitch adjusting rate.

property bw: float

Bandwidth parameter.

_generate_new_harmony(*agents: List[Agent]*) → *Agent*

It generates a new harmony.

Parameters

agents – List of agents.

Returns

A new agent (harmony) based on music generation process.

Return type

(*Agent*)

update(*space: Space, function: Function*) → None

Wraps Harmony Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

class `optimizer.optimizers.evolutionary.hs.IHS`(*params: Dict[str, Any] | None = None*)

An IHS class, inherited from HS.

This is the designed class to define IHS-related variables and methods.

References

M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation* (2007).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property PAR_min: float

Minimum pitch adjusting rate.

property PAR_max: float

Maximum pitch adjusting rate.

property bw_min: float

Minimum bandwidth parameter.

property bw_max: float

Maximum bandwidth parameter.

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Improved Harmony Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.

- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

class `optimizer.optimizers.evolutionary.hs.GHS`(*params: Dict[str, Any] | None = None*)

A GHS class, inherited from IHS.

This is the designed class to define GHS-related variables and methods.

References

M. Omran and M. Mahdavi. Global-best harmony search. Applied Mathematics and Computation (2008).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_generate_new_harmony(*agents: List[Agent]*) → *Agent*

It generates a new harmony.

Parameters

agents – List of agents.

Returns

A new agent (harmony) based on music generation process.

Return type

(*Agent*)

class `optimizer.optimizers.evolutionary.hs.SGHS`(*params: Dict[str, Any] | None = None*)

A SGHS class, inherited from HS.

This is the designed class to define SGHS-related variables and methods.

References

Q.-K. Pan, P. Suganthan, M. Tasgetiren and J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. Applied Mathematics and Computation (2010).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property HMCR: float

Harmony memory considering rate.

property PAR: float

Pitch adjusting rate.

property LP: int

Learning period.

property HMCRm: float

Mean harmony memory considering rate.

property PARM: float

Mean pitch adjusting rate.

property bw_min: float

Minimum bandwidth parameter.

property bw_max: float

Maximum bandwidth parameter.

property lp: int

Current learning period.

property HMCR_history: List[float]

Historical harmony memory considering rates.

property PAR_history: List[float]

Historical pitch adjusting rates.

compile(space: Space) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_generate_new_harmony(agents: List[Agent]) → Agent

It generates a new harmony.

Parameters

agents – List of agents.

Returns

A new agent (harmony) based on music generation process.

Return type

(Agent)

update(space: Space, function: Function, iteration: int, n_iterations: int) → None

Wraps Self-Adaptive Global-Best Harmony Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

class opytimizer.optimizers.evolutionary.hs.NGHS(params: Dict[str, Any] | None = None)

A NGHS class, inherited from HS.

This is the designed class to define NGHS-related variables and methods.

References

D. Zou, L. Gao, J. Wu and S. Li. Novel global harmony search algorithm for unconstrained problems. Neuro-computing (2010).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property pm: float

Mutation probability.

_generate_new_harmony(*best: Agent, worst: Agent*) → *Agent*

It generates a new harmony.

Parameters

- **best** – Best agent.
- **worst** – Worst agent.

Returns

A new agent (harmony) based on music generation process.

Return type

(*Agent*)

update(*space: Space, function: Function*) → None

Wraps Novel Global Harmony Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

class opytimizer.optimizers.evolutionary.hs.GOGHS(*params: Dict[str, Any] | None = None*)

A GOGHS class, inherited from NGHS.

This is the designed class to define GOGHS-related variables and methods.

References

Z. Guo, S. Wang, X. Yue and H. Yang. Global harmony search with generalized opposition-based learning. Soft Computing (2017).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_generate_opposition_harmony(*new_agent: Agent, agents: List[Agent]*) → *Agent*

It generates a new opposition-based harmony.

Parameters

- **new_agent** – Newly created agent.
- **agents** – List of agents.

Returns

A new agent (harmony) based on opposition generation process.

Return type

(*Agent*)

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Generalized Opposition Global-Best Harmony Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.2.9 opytimizer.optimizers.evolutionary.iwo

Invasive Weed Optimization.

class opytimizer.optimizers.evolutionary.iwo.**IWO**(*params*: *Dict[str, Any]* | *None = None*)

An IWO class, inherited from [Optimizer](#).

This is the designed class to define IWO-related variables and methods.

References

A. R. Mehrabian and C. Lucas. A novel numerical optimization algorithm inspired from weed colonization. Ecological informatics (2006).

__init__(*params*: *Dict[str, Any]* | *None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property min_seeds: **int**

Minimum number of seeds.

property max_seeds: **int**

Maximum number of seeds.

property e: **float**

Exponent used to calculate the Spatial Dispersal.

property final_sigma: **float**

Final standard deviation.

property init_sigma: **float**

Initial standard deviation.

property sigma: **float**

Standard deviation.

_spatial_dispersal(*iteration*: *int*, *n_iterations*: *int*) → None

Calculates the Spatial Dispersal coefficient (eq. 1).

Parameters

- **iteration** – Current iteration number.
- **n_iterations** – Maximum number of iterations.

_produce_offspring(*agent*: [Agent](#), *function*: [Function](#)) → [Agent](#)

Reproduces and flowers a seed into a new offspring.

Parameters

- **agent** – An agent instance to be reproduced.

- **function** – A Function object that will be used as the objective function.

Returns

An evolved offspring.

Return type

(*Agent*)

update(*space*: *Space*, *function*: *Function*, *iteration*: *int*, *n_iterations*: *int*) → *None*

Wraps Invasive Weed Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.2.10 opyimizer.optimizers.evolutionary.rra

Runner-Root Algorithm.

class opyimizer.optimizers.evolutionary.rra.**RRA**(*params*: *Dict[str, Any]* | *None* = *None*)

An RRA class, inherited from Optimizer.

This is the designed class to define RRA-related variables and methods.

References

F. Merrih-Bayat. The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. Applied Soft Computing (2015).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → *None*

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property d_runner: *int*

Length of runners.

property d_root: *float*

Length of roots.

property tol: *float*

Cost function tolerance.

property max_stall: *int*

Maximum number of stalls.

property n_stall: *int*

Current number of stalls.

property last_best_fit: *float*

Previous best fitness value.

_stalling_search(*daughters*: List[Agent], *function*: Function, *is_large*: bool | None = True) → None

Performs the stalling random larrge or small search (eq. 4 and 5).

Parameters

- **daughters** – Daughters.
- **function** – A Function object that will be used as the objective function.
- **is_large** – Whether to perform the large or small search.

_roulette_selection(*fitness*: List[float], *a*: float | None = 0.1) → int

Performs a roulette selection on the population (eq. 8).

Parameters

- **fitness** – A fitness list of every agent.
- **a** – Selection regularizer.

Returns

The selected index of the population.

Return type

(int)

update(*space*: Space, *function*: Function) → None

Wraps Runner-Root Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

An evolutionary package for all common opytimizer modules. It contains implementations of evolutionary-based opytimizers.

5.3 opytimizer.optimizers.misc

5.3.1 opytimizer.optimizers.misc.aoa

Arithmetic Optimization Algorithm.

class opytimizer.optimizers.misc.aoa.**AOA**(*params*: Dict[str, Any] | None = None)

An AOA class, inherited from Optimizer.

This is the designed class to define AOA-related variables and methods.

References

L. Abualigah et al. The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering (2021).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property a_min: float

Minimum accelerated function.

property a_max: float

Maximum accelerated function.

property alpha: float

Sensitive parameter.

property mu: float

Control parameter.

update(space: [Space](#), iteration: int, n_iterations: int) → None

Wraps Arithmetic Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.3.2 opyimizer.optimizers.misc.cem

Cross-Entropy Method.

class opyimizer.optimizers.misc.cem.**CEM**(params: Dict[str, Any] | None = None)

A CEM class, inherited from Optimizer.

This is the designed class to define CEM-related variables and methods.

References

R. Y. Rubinstein. Optimization of Computer simulation Models with Rare Events. European Journal of Operations Research (1997).

__init__(params: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_updates: int

Number of positions to employ in update formulae.

property alpha: float

Learning rate.

property mean: ndarray

Array of means.

property std: ndarray

Array of standard deviations.

compile(space: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_create_new_samples(agents: List[Agent], function: Function) → None

Creates new agents based on current mean and standard deviation.

Parameters

- **agents** (*list*) – List of agents.
- **function** – A Function object that will be used as the objective function.

_update_mean(updates: ndarray) → ndarray

Calculates and updates mean.

Parameters

updates – An array of updates' positions.

Returns

The new mean values.

Return type

(np.ndarray)

_update_std(updates: ndarray) → ndarray

Calculates and updates standard deviation.

Parameters

updates – An array of updates' positions.

Returns

The new standard deviation values.

Return type

(np.ndarray)

update(space: Space, function: Function) → None

Wraps Cross-Entropy Method over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.3.3 opyimizer.optimizers.misc.doa

Darcy Optimization Algorithm.

class opyimizer.optimizers.misc.doa.DOA(*params: Dict[str, Any] | None = None*)

A DOA class, inherited from Optimizer.

This is the designed class to define DOA-related variables and methods.

References

F. Demir et al. A survival classification method for hepatocellular carcinoma patients with chaotic Darcy optimization method based feature selection. Medical Hypotheses (2020).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property r: float

Chaos multiplier.

property chaotic_map: ndarray

Array of chaotic maps.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_calculate_chaotic_map(*lb: float, ub: float*) → float

Calculates the chaotic map (eq. 3).

Parameters

- **lb** – Lower bound value.
- **ub** – Upper bound value.

Returns

A new value for the chaotic map.

Return type

(float)

update(*space: Space*) → None

Wraps Darcy Optimization Algorithm over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.3.4 opytimizer.optimizers.misc.gs

Grid-Search.

class opytimizer.optimizers.misc.gs.**GS**(*params: Dict[str, Any] | None = None*)

A GS class, inherited from Optimizer.

This is the designed class to define grid search-related variables and methods.

References

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. Journal of machine learning research (2012).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

5.3.5 opytimizer.optimizers.misc.hc

Hill-Climbing.

class opytimizer.optimizers.misc.hc.**HC**(*params: Dict[str, Any] | None = None*)

An HC class, inherited from Optimizer.

This is the designed class to define HC-related variables and methods.

References

S. Skiena. The Algorithm Design Manual (2010).

__init__(*params*: *Dict[str, Any]* | *None = None*) → *None*

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property r_mean: **float**

Mean of noise distribution.

property r_var: **float**

Variance of noise distribution.

update(*space*: *Space*) → *None*

Wraps Hill Climbing over all agents and variables (p. 252).

Parameters

space – Space containing agents and update-related information.

5.3.6 opyimizer.optimizers.misc.nds

Non-Dominated Sorting.

class opyimizer.optimizers.misc.nds.**NDS**(*params*: *Dict[str, Any]* | *None = None*)

An NDS class, inherited from Optimizer.

This is the designed class to define NDS-related variables and methods.

References

P. Godfrey, R. Shipley and J. Gryz. Algorithms and Analyses for Maximal Vector Computation. The VLDB Journal (2007).

__init__(*params*: *Dict[str, Any]* | *None = None*) → *None*

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_pareto_points: **int**

Number of points in the frontier.

property count: **ndarray**

Array of domination counts.

property set: **ndarray**

Array of dominating set.

property status: **ndarray**

Array of pareto status.

compile(*space*: *Space*) → *None*

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_compare_domination(*agent_i*: [Agent](#), *agent_j*: [Agent](#)) → bool

Calculates whether *i* dominates *j*.

Parameters

- **agent_i** – Agent *i*.
- **agent_j** – Agent *j*.

Returns

Boolean indicating whether *i* dominated *j* or not.

Return type

(bool)

update(*space*: [Space](#)) → None

Wraps Non-Dominated Sorting over all agents and variables.

Parameters

space – Space containing agents and update-related information.

An evolutionary package for all common opyoptimizer modules. It contains implementations of miscellaneous-based optimizers.

5.4 opyoptimizer.optimizers.population

5.4.1 opyoptimizer.optimizers.population.aeo

Artificial Ecosystem-based Optimization.

class opyoptimizer.optimizers.population.aeo.**AEO**(*params*: *Dict[str, Any] | None = None*)

An AEO class, inherited from Optimizer.

This is the designed class to define AEO-related variables and methods.

References

W. Zhao, L. Wang and Z. Zhang. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Computing and Applications (2019).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_production(*agent*: [Agent](#), *best_agent*: [Agent](#), *iteration*: int, *n_iterations*: int) → [Agent](#)

Performs the producer update (eq. 1).

Parameters

- **agent** – Current agent.
- **best_agent** – Best agent.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

An updated producer.

Return type*(Agent)***_herbivore_consumption**(*agent: Agent, producer: Agent, C: float*) → *Agent*

Performs the consumption update by a herbivore (eq. 6).

Parameters

- **agent** – Current agent.
- **producer** – Producer agent.
- **C** – Consumption factor.

Returns

An updated consumption by a herbivore.

_omnivore_consumption(*agent: Agent, producer: Agent, consumer: Agent, C: float*) → *Agent*

Performs the consumption update by an omnivore (eq. 8)

Parameters

- **agent** – Current agent.
- **producer** – Producer agent.
- **consumer** – Consumer agent.
- **C** – Consumption factor.

Returns

An updated consumption by an omnivore.

Return type*(Agent)***_carnivore_consumption**(*agent: Agent, consumer: Agent, C: float*) → *Agent*

Performs the consumption update by a carnivore (eq. 7).

Parameters

- **agent** – Current agent.
- **consumer** – Consumer agent.
- **C** – Consumption factor.

Returns

An updated consumption by a carnivore.

Return type*(Agent)***_update_composition**(*agents: List[Agent], best_agent: Agent, function: Function, iteration: int, n_iterations: int*) → *None*

Wraps production and consumption updates over all agents and variables (eq. 1-8).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.

- **n_iterations** – Maximum number of iterations.

_update_decomposition(*agents*: List[Agent], *best_agent*: Agent, *function*: Function) → None

Wraps decomposition updates over all agents and variables (eq. 9).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.
- **function** – A Function object that will be used as the objective function.

update(*space*: Space, *function*: Function, *iteration*: int, *n_iterations*: int) → None

Wraps Artificial Ecosystem-based Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.2 opyimizer.optimizers.population.ao

Aquila Optimizer.

class opyimizer.optimizers.population.ao.**AO**(*params*: Dict[str, Any] | None = None)

An AO class, inherited from Optimizer.

This is the designed class to define AO-related variables and methods.

References

L. Abualigah et al. Aquila Optimizer: A novel meta-heuristic optimization Algorithm. Computers & Industrial Engineering (2021).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

First exploitation adjustment coefficient.

property delta: float

Second exploitation adjustment coefficient.

property n_cycles: int

Number of cycles.

property U: float

Cycle regularizer.

property w: float

Angle regularizer.

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Aquila Optimizer over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.3 opytimizer.optimizers.population.coa

Coyote Optimization Algorithm.

class opytimizer.optimizers.population.coa.COA(*params*: *Dict[str, Any]* | *None* = *None*)

A COA class, inherited from Optimizer.

This is the designed class to define COA-related variables and methods.

References

J. Pierezan and L. Coelho. Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems. IEEE Congress on Evolutionary Computation (2018).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_p: *int*

Number of packs.

property n_c: *int*

Number of coyotes per pack.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_get_agents_from_pack(*agents*: *List[Agent]*, *index*: *int*) → *List[Agent]*

Gets a set of agents from a specified pack.

Parameters

- **agents** – List of agents.
- **index** – Index of pack.

Returns

A sorted list of agents that belongs to the specified pack.

Return type

(*List[Agent]*)

_transition_packs(*agents: List[Agent]*) → None

Transits coyotes between packs (eq. 4).

Parameters

agents – List of agents.

update(*space: Space, function: Function*) → None

Wraps Coyote Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.4.4 opyimizer.optimizers.population.epo

Emperor Penguin Optimizer.

class opyimizer.optimizers.population.epo.**EPO**(*params: Dict[str, Any] | None = None*)

An EPO class, inherited from Optimizer.

This is the designed class to define EPO-related variables and methods.

References

G. Dhiman and V. Kumar. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowledge-Based Systems (2018).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property f: float

Exploration control parameter.

property l: float

Exploitation control parameter.

update(*space: Space, iteration: int, n_iterations: int*) → None

Wraps Emperor Penguin Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.5 opyimizer.optimizers.population.gco

Germinal Center Optimization.

class opyimizer.optimizers.population.gco.**GCO**(*params: Dict[str, Any] | None = None*)

A GCO class, inherited from Optimizer.

This is the designed class to define GCO-related variables and methods.

References

C. Villaseñor et al. Germinal center optimization algorithm. International Journal of Computational Intelligence Systems (2018).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property CR: float

Cross-ratio parameter.

property F: float

Mutation factor.

property life: ndarray

Array of lives.

property counter: ndarray

Array of counters.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_mutate_cell(*agent: Agent, alpha: Agent, beta: Agent, gamma: Agent*) → *Agent*

Mutates a new cell based on distinct cells (alg. 2).

Parameters

- **agent** – Current agent.
- **alpha** – 1st picked agent.
- **beta** – 2nd picked agent.
- **gamma** – 3rd picked agent.

Returns

A mutated cell.

Return type

(*Agent*)

_dark_zone(*agents: List[Agent], function: Function*) → None

Performs the dark-zone update process (alg. 1).

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.

_light_zone(*agents: List[Agent]*) → None

Performs the light-zone update process (alg. 1).

Parameters

agents – List of agents.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Germinal Center Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.4.6 opyoptimizer.optimizers.population.gwo

Grey Wolf Optimizer.

class opyoptimizer.optimizers.population.gwo.**GWO**(*params*: *Dict[str, Any] | None = None*)

A GWO class, inherited from Optimizer.

This is the designed class to define GWO-related variables and methods.

References

S. Mirjalili, S. Mirjalili and A. Lewis. Grey Wolf Optimizer. Advances in Engineering Software (2014).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_calculate_coefficients(*a*: *float*) → *Tuple[float, float]*

Calculates the mathematical coefficients.

Parameters

a – Linear constant.

Returns

Both *A* and *C* coefficients.

Return type

Tuple[float, float]

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Grey Wolf Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.7 opyoptimizer.optimizers.population.hho

Harris Hawks Optimization.

class opyoptimizer.optimizers.population.hho.**HHO**(*params*: *Dict[str, Any] | None = None*)

An HHO class, inherited from Optimizer.

This is the designed class to define HHO-related variables and methods.

References

A. Heidari et al. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems (2019).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_calculate_initial_coefficients(*iteration: int, n_iterations: int*) → Tuple[float, float]

Calculates the initial coefficients, i.e., energy and jump's strength.

Parameters

- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

Absolute value of energy and jump's strength.

Return type

(Tuple[float, float])

_exploration_phase(*agents: List[Agent], current_agent: Agent, best_agent: Agent*) → ndarray

Performs the exploration phase.

Parameters

- **agents** – List of agents.
- **current_agent** – Current agent to be updated (or not).
- **best_agent** – Best population's agent.

Returns

A location vector containing the updated position.

Return type

(np.ndarray)

_exploitation_phase(*energy: float, jump: float, agents: List[Agent], current_agent: Agent, best_agent: Agent, function: Function*) → ndarray

Performs the exploitation phase.

Parameters

- **energy** – Energy coefficient.
- **jump** – Jump's strength.
- **agents** – List of agents.
- **current_agent** – Current agent to be updated (or not).
- **best_agent** – Best population's agent.
- **function** – A function object.

Returns

A location vector containing the updated position.

Return type

(np.ndarray)

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Harris Hawks Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.8 opyimizer.optimizers.population.loa

Lion Optimization Algorithm.

class opyimizer.optimizers.population.loa.**Lion**(*n_variables*: *int*, *n_dimensions*: *int*, *lower_bound*: *List* | *Tuple* | *ndarray*, *upper_bound*: *List* | *Tuple* | *ndarray*, *position*: *ndarray*, *fit*: *float*)

A Lion class complements its inherited parent with additional information needed by the Lion Optimization Algorithm.

__init__(*n_variables*: *int*, *n_dimensions*: *int*, *lower_bound*: *List* | *Tuple* | *ndarray*, *upper_bound*: *List* | *Tuple* | *ndarray*, *position*: *ndarray*, *fit*: *float*) → None

Initialization method.

Parameters

- **n_variables** – Number of decision variables.
- **n_dimensions** – Number of dimensions.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **position** – Position array.
- **fit** – Fitness value.

property best_position: **ndarray**

N-dimensional array of best positions.

property p_fit: **float**

Previous fitness value.

property nomad: **bool**

Whether lion is nomad or not.

Type

bool

property female: **bool**

Whether lion is female or not.

property pride: **int**

Index of pride.

property group: **int**

Index of hunting group.

class `optimizer.optimizers.population.loa.LOA`(*params: Dict[str, Any] | None = None*)

An LOA class, inherited from Optimizer.

This is the designed class to define LOA-related variables and methods.

References

M. Yazdani and F. Jolai. Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. Journal of Computational Design and Engineering (2016).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property N: float

Percentage of nomad lions.

property P: int

Number of prides.

property S: float

Percentage of female lions.

property R: float

Percentage of roaming lions.

property I: float

Immigrate rate.

property Ma: float

Mating probability.

property Mu: float

Mutation probability.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_get_nomad_lions(*agents: List[Lion]*) → List[Lion]

Gets all nomad lions.

Parameters

agents – Agents.

Returns

A list of nomad lions.

Return type

(List[Lion])

_get_pride_lions(*agents: List[Lion]*) → List[List[Lion]]

Gets all non-nomad (pride) lions.

Parameters

agents – Agents.

Returns

A list of lists, where each one indicates a particular pride with its lions.

Return type

(List[List[Lion]])

_hunting(*prides: List[Lion], function: Function*) → None

Performs the hunting procedure (s. 2.2.2).

Parameters

- **prides** – List of prides holding their corresponding lions.
- **function** – A Function object that will be used as the objective function.

_moving_safe_place(*prides: List[Lion]*) → None

Move prides to safe locations (s. 2.2.3).

Parameters

prides – List of prides holding their corresponding lions.

_roaming(*prides: List[Lion], function: Function*) → None

Performs the roaming procedure (s. 2.2.4).

Parameters

- **prides** – List of prides holding their corresponding lions.
- **function** – A Function object that will be used as the objective function.

_mating_operator(*agent: List[Lion], males: List[Lion], function: Function*) → Tuple[Lion, Lion]

Wraps the mating operator.

Parameters

- **agent** – Current agent.
- **males** – List of males that will be breed.
- **function** – A Function object that will be used as the objective function.

Returns

A pair of offsprings that resulted from mating.

Return type

(Tuple[Lion, Lion])

_mating(*prides: List[Lion], function: Function*) → Lion

Generates offsprings from mating (s. 2.2.5).

Parameters

- **prides** – List of prides holding their corresponding lions.
- **function** – A Function object that will be used as the objective function.

Returns

Cubs generated from the mating procedure.

Return type

(Lion)

`_defense`(*nomads*: List[Lion], *prides*: List[List[Lion]], *cubs*: List[Lion]) → Tuple[List[Lion], List[List[Lion]]]

Performs the defense procedure (s. 2.2.6).

Parameters

- **nomads** – Nomad lions.
- **prides** – List of prides holding their corresponding lions.
- **cubs** – List of cubs holding their corresponding lions.

Returns

Both updated nomad and pride lions.

Return type

(Tuple[List[Lion], List[List[Lion]]])

`_nomad_roaming`(*nomads*: List[Lion], *function*: Function) → None

Performs the roaming procedure for nomad lions (s. 2.2.4).

Parameters

- **nomads** – Nomad lions.
- **function** – A Function object that will be used as the objective function.

`_nomad_mating`(*nomads*: List[Lion], *function*: Function) → List[Lion]

Generates offsprings from nomad lions mating (s. 2.2.5).

Parameters

- **nomads** – Nomad lions.
- **function** – A Function object that will be used as the objective function.

Returns

Updated nomad lions.

Return type

(List[Lion])

`_nomad_attack`(*nomads*: List[Lion], *prides*: List[List[Lion]]) → Tuple[List[Lion], List[List[Lion]]]

Performs the nomad's attacking procedure (s. 2.2.6).

Parameters

- **nomads** – Nomad lions.
- **prides** – List of prides holding their corresponding lions.

Returns

Both updated nomad and pride lions.

Return type

(Tuple[List[Lion], List[List[Lion]]])

`_migrating`(*nomads*: List[Lion], *prides*: List[List[Lion]]) → Tuple[List[Lion], List[List[Lion]]]

Performs the nomad's migration procedure (s. 2.2.7).

Parameters

- **nomads** – Nomad lions.
- **prides** – List of prides holding their corresponding lions.

Returns

Both updated nomad and pride lions.

Return type

(Tuple[List[Lion], List[List[Lion]]])

_equilibrium(nomads: List[Lion], prides: List[List[Lion]], n_agents: List[Agent]) → Tuple[List[Lion], List[List[Lion]]

Performs the population's equilibrium procedure (s. 2.2.8).

Parameters

- **nomads** – Nomad lions.
- **prides** – List of prides holding their corresponding lions.

Returns

Both updated nomad and pride lions.

Return type

(Tuple[List[Lion], List[List[Lion]]])

_check_prides_for_males(prides: List[List[Lion]]) → None

Checks if there is at least one male per pride.

Parameters

prides – List of prides holding their corresponding lions.

update(space: Space, function: Function) → None

Wraps Lion Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.4.9 opyoptimizer.optimizers.population.osa

Owl Search Algorithm.

class opyoptimizer.optimizers.population.osa.OSA(params: Dict[str, Any] | None = None)

An OSA class, inherited from Optimizer.

This is the designed class to define OSA-related variables and methods.

References

M. Jain, S. Maurya, A. Rani and V. Singh. Owl search algorithm: A novel nature-inspired heuristic paradigm for global optimization. Journal of Intelligent & Fuzzy Systems (2018).

__init__(params: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property beta: float

Exploration intensity.

update(*space*: [Space](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Owl Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.10 opytizer.optimizers.population.ppa

Parasitism-Predation Algorithm.

class opytizer.optimizers.population.ppa.PPA(*params*: *Dict[str, Any] | None = None*)

A PPA class, inherited from Optimizer.

This is the designed class to define PPA-related variables and methods.

References

A. Mohamed et al. Parasitism – Predation algorithm (PPA): A novel approach for feature selection. Ain Shams Engineering Journal (2020).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property velocity: **ndarray**

Array of velocities.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_calculate_population(*n_agents*: *int*, *iteration*: *int*, *n_iterations*: *int*) → *Tuple[int, int, int]*

Calculates the number of crows, cats and cuckoos.

Parameters

- **n_agents** – Number of agents.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

The number of crows, cats and cuckoos.

Return type

(*Tuple[int, int, int]*)

_nesting_phase(*space*: [Space](#), *n_crows*: *int*)

Performs the nesting phase using the current number of crows.

Parameters

- **space** – Space containing agents and update-related information.

- **n_crows** – Number of crows.

_parasitism_phase(*space*: [Space](#), *n_crows*: *int*, *n_cuckoos*: *int*, *iteration*: *int*, *n_iterations*: *int*)

Performs the parasitism phase using the current number of cuckoos.

Parameters

- **space** – Space containing agents and update-related information.
- **n_crows** – Number of crows.
- **n_cuckoos** – Number of cuckoos.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

_predation_phase(*space*: [Space](#), *n_crows*: *int*, *n_cuckoos*: *int*, *n_cats*: *int*, *iteration*: *int*, *n_iterations*: *int*)
→ None

Performs the predation phase using the current number of cats.

Parameters

- **space** – Space containing agents and update-related information.
- **n_crows** – Number of crows.
- **n_cuckoos** – Number of cuckoos.
- **n_cats** – Number of cats.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

update(*space*: [Space](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Parasitism-Predation Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.4.11 opyoptimizer.optimizers.population.pvs

Passing Vehicle Search.

class opyoptimizer.optimizers.population.pvs.**PVS**(*params*: *Dict[str, Any]* | *None* = *None*)

A PVS class, inherited from [Optimizer](#).

This is the designed class to define PVS-related variables and methods.

References

P. Savsani and V. Savsani. Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling* (2016).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

- **params** – Contains key-value parameters to the meta-heuristics.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Passing Vehicle Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.4.12 optimizer.optimizers.population.rfo

Red Fox Optimization.

class optimizer.optimizers.population.rfo.**RFO**(*params*: *Dict[str, Any] | None = None*)

A RFO class, inherited from Optimizer.

This is the designed class to define RFO-related variables and methods.

References

D. Polap and M. Woźniak. Red fox optimization algorithm. Expert Systems with Applications (2021).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property phi: float

Observation angle.

property theta: float

Weather condition.

property p_replacement: float

Percentual of foxes replacement.

property n_replacement: int

Number of foxes to be replaced.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_relocation(*agent*: [Agent](#), *best_agent*: [Agent](#), *function*: [Function](#)) → None

Performs the fox relocation procedure.

Parameters

- **agent** – Current agent.
- **best_agent** – Best agent.
- **function** – A Function object that will be used as the objective function.

_noticing(*agent*: [Agent](#), *function*: [Function](#), *alpha*: float) → None

Performs the fox noticing procedure.

Parameters

- **agent** – Current agent.

- **function** – A Function object that will be used as the objective function.
- **alpha** – Scaling parameter.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Red Fox Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

An evolutionary package for all common opyoptimizer modules. It contains implementations of population-based optimizers.

5.5 opyoptimizer.optimizers.science

5.5.1 opyoptimizer.optimizers.science.aig

Algorithm of the Innovative Gunner.

class opyoptimizer.optimizers.science.aig.**AIG**(*params*: *Dict[str, Any]* | *None* = *None*)

An AIG class, inherited from [Optimizer](#).

This is the designed class to define AIG-related variables and methods.

References

P. Pijarski and P. Kacejko. A new metaheuristic optimization method: the algorithm of the innovative gunner (AIG). Engineering Optimization (2019).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

First maximum correction angle.

property beta: float

Second maximum correction angle.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Algorithm of the Innovative Gunner over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.5.2 opyoptimizer.optimizers.science.aso

Atom Search Optimization.

class opyoptimizer.optimizers.science.aso.**ASO**(*params*: *Dict[str, Any]* | *None* = *None*)

An ASO class, inherited from [Optimizer](#).

This is the designed class to define ASO-related variables and methods.

References

W. Zhao, L. Wang and Z. Zhang. A novel atom search optimization for dispersion coefficient estimation in groundwater. Future Generation Computer Systems (2019).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

Depth weight.

property beta: float

Multiplier weight.

property velocity: ndarray

Array of velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_calculate_mass(*agents: List[Agent]*) → List[float]

Calculates the atoms' masses (eq. 17 and 18).

Parameters

agents – List of agents.

Returns

A list holding the atoms' masses.

Return type

(List[float])

_calculate_potential(*agent: Agent, K_agent: Agent, average: ndarray, iteration: int, n_iterations: int*) → None

Calculates the potential of an agent based on its neighbour and average positioning.

Parameters

- **agent** – Agent to have its potential calculated.
- **K_agent** – Neighbour agent.
- **average** – Array of average positions.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

_calculate_acceleration(*agents: List[Agent], best_agent: Agent, mass: ndarray, iteration: int, n_iterations: int*) → ndarray

Calculates the atoms' acceleration.

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.

- **mass** – Array of masses.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

An array holding the atoms' acceleration.

Return type

(np.ndarray)

update(*space*: [Space](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Atom Search Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.3 opyoptimizer.optimizers.science.bh

Black Hole.

class opyoptimizer.optimizers.science.bh.**BH**(*params*: *Dict[str, Any]* | *None* = *None*)

A BH class, inherited from Optimizer.

This is the designed class to define BH-related variables and methods.

References

A. Hatamlou. Black hole: A new heuristic optimization approach for data clustering. Information Sciences (2013).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_update_position(*agents*: *List[Agent]*, *best_agent*: [Agent](#), *function*: [Function](#)) → float

It updates every star position and calculates their event's horizon cost (eq. 3).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.
- **function** – A function object.

Returns

The cost of the event horizon.

Return type

(float)

_event_horizon(*agents*: *List[Agent]*, *best_agent*: [Agent](#), *cost*: *float*) → None

It calculates the stars' crossing an event horizon (eq. 4).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.
- **cost** – The event’s horizon cost.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Black Hole over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.5.4 opytimizer.optimizers.science.efo

Electromagnetic Field Optimization.

class opytimizer.optimizers.science.efo.EFO(*params*: *Dict[str, Any] | None = None*)

An EFO class, inherited from Optimizer.

This is the designed class to define EFO-related variables and methods.

References

H. Abedinpourshotorban et al. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. Swarm and Evolutionary Computation (2016).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property positive_field: float

Positive field proportion.

property negative_field: float

Negative field proportion.

property ps_ratio: float

Probability of selecting eletromagnets.

property r_ratio: float

Probability of selecting a random eletromagnet.

property phi: float

Golden ratio.

property RI: float

Eletromagnetic index.

_calculate_indexes(*n_agents*: int) → Tuple[int, int, int]

Calculates the indexes of positive, negative and neutral particles.

Parameters

n_agents – Number of agents in the space.

Returns

Positive, negative and neutral particles’ indexes.

Return type

(Tuple[int, int, int])

update(*space*: Space, *function*: Function) → None

Wraps Electromagnetic Field Optimization over all agents and variables (eq. 1-4).

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.5.5 opytizer.optimizers.science.eo

Equilibrium Optimizer.

class opytizer.optimizers.science.eo.EO(*params*: Dict[str, Any] | None = None)

An EO class, inherited from Optimizer.

This is the designed class to define EO-related variables and methods.

References

A. Faramarzi et al. Equilibrium optimizer: A novel optimization algorithm. Knowledge-Based Systems (2020).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters**params** – Contains key-value parameters to the meta-heuristics.**property a1**: float

Exploration constant.

property a2: float

Exploitation constant.

property GP: float

Generation probability.

property V: float

Velocity.

property C: List[Agent]

Concentrations (agents).

compile(*space*: Space) → None

Compiles additional information that is used by this optimizer.

Parameters**space** – A Space object containing meta-information.**_calculate_equilibrium**(*agents*: List[Agent]) → None

Calculates the equilibrium concentrations.

Parameters**agents** – List of agents.

_average_concentration(*function*: [Function](#)) → *Agent*

Averages the concentrations.

Parameters

function – A Function object that will be used as the objective function.

Returns

Averaged concentration.

Return type

(*Agent*)

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Equilibrium Optimizer over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.6 opytizer.optimizers.science.esa

Electro-Search Algorithm.

class opytizer.optimizers.science.esa.**ESA**(*params*: *Dict[str, Any]* | *None* = *None*)

An ESA class, inherited from Optimizer.

This is the designed class to define ES-related variables and methods.

References

A. Tabari and A. Ahmad. A new optimization method: Electro-Search algorithm. Computers & Chemical Engineering (2017).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_electrons: **int**

Number of electrons per atom.

property D: **ndarray**

Orbital radius.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps EElectro-Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.

- **function** – A Function object that will be used as the objective function.

5.5.7 opyimizer.optimizers.science.gsa

Gravitational Search Algorithm.

class opyimizer.optimizers.science.gsa.**GSA**(*params: Dict[str, Any] | None = None*)

A GSA class, inherited from Optimizer.

This is the designed class to define GSA-related variables and methods.

References

E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi. GSA: a gravitational search algorithm. Information Sciences (2009).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property G: float

Initial gravity.

property velocity: ndarray

Array of velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_calculate_mass(*agents: List[Agent]*) → float

Calculates agents' mass (eq. 16).

Parameters

agents – List of agents.

Returns

The agents' mass.

Return type

(float)

_calculate_force(*agents: List[Agent], mass: ndarray, gravity: float*) → float

Calculates agents' force (eq. 7-9).

Parameters

- **agents** – List of agents.
- **mass** – An array of agents' mass.
- **gravity** – Current gravity value.

Returns

The attraction force between all agents.

Return type

(float)

update(*space*: [Space](#), *iteration*: *int*) → None

Wraps Gravitational Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.

5.5.8 opyimizer.optimizers.science.hgso

Henry Gas Solubility Optimization.

class opyimizer.optimizers.science.hgso.**HGSO**(*params*: *Dict[str, Any]* | *None* = *None*)

An HGSO class, inherited from [Optimizer](#).

This is the designed class to define HGSO-related variables and methods.

References

F. Hashim et al. Henry gas solubility optimization: A novel physics-based algorithm. Future Generation Computer Systems (2019).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_clusters: int

Number of clusters.

property l1: float

Henry’s coefficient constant.

property l2: int

Partial pressure constant.

property l3: float

Constant.

property alpha: float

Influence of gases.

property beta: float

Gas constant.

property K: float

Solubility constant.

property coefficient: ndarray

Array of coefficients.

property pressure: ndarray

Array of pressures.

property constant: ndarray

Array of constants.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_update_position(*agent*: [Agent](#), *cluster_agent*: [Agent](#), *best_agent*: [Agent](#), *solubility*: *float*) → ndarray

Updates the position of a single gas (eq. 10).

Parameters

- **agent** – Current agent.
- **cluster_agent** – Best cluster’s agent.
- **best_agent** – Best agent.
- **solubility** – Solubility for current agent.

Returns

An updated position.

Return type

(np.ndarray)

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Henry Gas Solubility Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.9 opyoptimizer.optimizers.science.lsa

Lightning Search Algorithm.

class opyoptimizer.optimizers.science.lsa.**LSA**(*params*: *Dict[str, Any]* | *None = None*)

An LSA class, inherited from Optimizer.

This is the designed class to define LSA-related variables and methods.

References

H. Shareef, A. Ibrahim and A. Mutlag. Lightning search algorithm. Applied Soft Computing (2015).

__init__(*params*: *Dict[str, Any]* | *None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property max_time: *int*

Maximum channel time.

property E: *float*

Initial energy.

property p_fork: float

Probability of forking.

property time: int

Channel time.

property direction: ndarray

Array of directions.

compile(space: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_update_direction(agent: [Agent](#), function: [Function](#)) → None

Updates the direction array by shaking agent's direction.

Parameters

- **agent** – An agent instance.
- **function** – A Function object that will be used as the objective function.

_update_position(agent: [Agent](#), best_agent: [Agent](#), function: [Function](#), energy: float) → None

Updates agent's position.

Parameters

- **agent** – An agent instance.
- **best_agent** – A best agent instance.
- **function** – A Function object that will be used as the objective function.
- **energy** – Current energy value.

update(space: [Space](#), function: [Function](#), iteration: int, n_iterations: int) → None

Wraps Lightning Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.10 opytimizer.optimizers.science.moa

Magnetic Optimization Algorithm.

class opytimizer.optimizers.science.moa.MOA(params: Dict[str, Any] | None = None)

An MOA class, inherited from Optimizer.

This is the designed class to define MOA-related variables and methods.

References

M.-H. Tayarani and M.-R. Akbarzadeh. Magnetic-inspired optimization algorithms: Operators and structures. Swarm and Evolutionary Computation (2014).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

Particle movement first constant.

property rho: float

Particle movement second constant.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space*) → None

Wraps Magnetic Optimization Algorithm over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.5.11 opytimizer.optimizers.science.mvo

Multi-Verse Optimizer.

class opytimizer.optimizers.science.mvo.**MVO**(*params: Dict[str, Any] | None = None*)

A MVO class, inherited from Optimizer.

This is the designed class to define MVO-related variables and methods.

References

S. Mirjalili, S. M. Mirjalili and A. Hatamlou. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications (2016).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property WEP_min: float

Minimum Wormhole Existence Probability.

property WEP_max: float

Maximum Wormhole Existence Probability.

property p: float

Exploitation accuracy.

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → *None*

Wraps Multi-Verse Optimizer over all agents and variables (eq. 3.1-3.4).

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.12 opytimizer.optimizers.science.sa

Simulated Annealing.

class opytimizer.optimizers.science.sa.**SA**(*params*: *Dict[str, Any]* | *None* = *None*)

A SA class, inherited from Optimizer.

This is the designed class to define SA-related variables and methods.

References

A. Khachaturyan, S. Semenovsovskaya and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. Acta Crystallographica (1981).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → *None*

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property T: **float**

System's temperature.

property beta: **float**

Temperature decay.

update(*space*: [Space](#), *function*: [Function](#)) → *None*

Wraps Simulated Annealing over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A function object.

5.5.13 opytimizer.optimizers.science.teo

Thermal Exchange Optimization.

class opytimizer.optimizers.science.teo.**TEO**(*params*: *Dict[str, Any]* | *None* = *None*)

A TEO class, inherited from Optimizer.

This is the designed class to define TEO-related variables and methods.

References

A. Kaveh and A. Dadras. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software* (2017).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c1: bool

Random step size control.

property c2: bool

Randomness control.

property pro: float

Cooling parameter.

property n_TM: int

Size of thermal memory.

property TM: List[Agent]

Thermal memory.

property environment: List[Agent]

Environmental population.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space, iteration: int, n_iterations: int*) → None

Wraps Thermal Exchange Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.14 opytimizer.optimizers.science.two

Tug Of War Optimization.

class opytimizer.optimizers.science.two.TWO(*params: Dict[str, Any] | None = None*)

A TWO class, inherited from Optimizer.

This is the designed class to define TWO-related variables and methods.

References

A. Kaveh. Tug of War Optimization. *Advances in Metaheuristic Algorithms for Optimal Design of Structures* (2016).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property mu_s: float

Static friction coefficient.

property mu_k: float

Kinematic friction coefficient.

property delta_t: float

Time displacement.

property alpha: float

Speed constant.

property beta: float

Scaling factor.

_constraint_handle(*agents: List[Agent], best_agent: Agent, function: Function, iteration: int*) → None

Performs the constraint handling procedure (eq. 11).

Parameters

- **agents** (*list*) – List of agents.
- **best_agent** (*Agent*) – Global best agent.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Tug of War Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.15 opytimizer.optimizers.science.wca

Water Cycle Algorithm.

class opytimizer.optimizers.science.wca.**WCA**(*params: Dict[str, Any] | None = None*)

A WCA class, inherited from Optimizer.

This is the designed class to define WCA-related variables and methods.

References

H. Eskandar. Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures (2012).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property nsr: float

Number of rivers summed with a single sea.

property d_max: float

Maximum evaporation condition.

property flows: ndarray

Array of flows.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_flow_intensity(*agents: List[Agent]*) → None

Calculates the intensity of each possible flow (eq. 6).

Parameters

agents – List of agents.

_raining_process(*agents: List[Agent], best_agent: Agent*) → None

Performs the raining process (eq. 11-12).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.

_update_stream(*agents: List[Agent], function: Function*) → None

Updates every stream position (eq. 8).

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.

_update_river(*agents: List[Agent], best_agent: Agent, function: Function*) → None

Updates every river position (eq. 9).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.
- **function** – A Function object that will be used as the objective function.

update(*space: Space, function: Function, n_iterations: int*) → None

Wraps Water Cycle Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

- **n_iterations** – Maximum number of iterations.

5.5.16 opyoptimizer.optimizers.science.wdo

Wind Driven Optimization.

class opyoptimizer.optimizers.science.wdo.**WDO**(*params: Dict[str, Any] | None = None*)

A WDO class, inherited from Optimizer.

This is the designed class to define WDO-related variables and methods.

References

Z. Bayraktar et al. The wind driven optimization technique and its application in electromagnetics. IEEE transactions on antennas and propagation (2013).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property v_max: float

Maximum velocity.

property alpha: float

Friction coefficient.

property g: float

Gravitational force coefficient.

property c: float

Coriolis force.

property RT: float

Pressure constant.

property velocity: ndarray

Array of velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space, function: Function*) → None

Wraps Wind Driven Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A function object.

5.5.17 opyoptimizer.optimizers.science.weo

Water Evaporation Optimization.

class `opyoptimizer.optimizers.science.weo.WEO`(*params: Dict[str, Any] | None = None*)

A WEO class, inherited from Optimizer.

This is the designed class to define WEO-related variables and methods.

References

A. Kaveh and T. Bakhshpoori. Water Evaporation Optimization: A novel physically inspired optimization algorithm. Computers & Structures (2016).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property E_min: float

Minimum substrate energy.

property E_max: float

Maximum substrate energy.

property theta_min: float

Minimum contact angle.

property theta_max: float

Maximum contact angle.

_evaporation_flux(*theta: float*) → float

Calculates the evaporation flux (eq. 7).

Parameters

theta – Radian-based angle.

Returns

Evaporation flux.

Return type

(float)

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Water Evaporation Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.5.18 opyoptimizer.optimizers.science.wwo

Water Wave Optimization.

class `opyoptimizer.optimizers.science.wwo.WWO`(*params: Dict[str, Any] | None = None*)

A WWO class, inherited from Optimizer.

This is the designed class to define WWO-related variables and methods.

References

Y.-J. Zheng. Water wave optimization: A new nature-inspired metaheuristic. Computers & Operations Research (2015).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property h_max: int

Maximum wave height.

property alpha: float

Wave length reduction coefficient.

property beta: float

Breaking coefficient.

property k_max: int

Maximum number of breakings.

property height: ndarray

Array of heights.

property length: ndarray

Array of lengths.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_propagate_wave(*agent: Agent, function: Function, index: int*) → Agent

Propagates wave into a new position (eq. 6).

Parameters

- **agent** – Current wave.
- **function** – A function object.
- **index** – Index of wave length.

Returns

Propagated wave.

Return type

(Agent)

_refract_wave(*agent: Agent, best_agent: Agent, function: Function, index: int*) → Tuple[float, float]

Refract wave into a new position (eq. 8-9).

Parameters

- **agent** – Agent to be refracted.
- **best_agent** – Global best agent.
- **function** – A function object.
- **index** – Index of wave length.

Returns

New height and length values.

Return type

(Tuple[float, float])

_break_wave(*wave*: [Agent](#), *function*: [Function](#), *j*: *int*) → [Agent](#)

Breaks current wave into a new one (eq. 10).

Parameters

- **wave** – Wave to be broken.
- **function** – A function object.
- **j** – Index of dimension to be broken.

Returns

Broken wave.

Return type

([Agent](#))

_update_wave_length(*agents*: List[[Agent](#)]) → None

Updates the wave length of current population.

Parameters

agents – List of agents.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Water Wave Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A function object.

An evolutionary package for all common opyoptimizer modules. It contains implementations of science-based optimizers.

5.6 opyoptimizer.optimizers.social

5.6.1 opyoptimizer.optimizers.social.bso

Brain Storm Optimization.

class opyoptimizer.optimizers.social.bso.**BSO**(*params*: Dict[str, Any] | None = None)

A BSO class, inherited from Optimizer.

This is the designed class to define BSO-related variables and methods.

References

Y. Shi. Brain Storm Optimization Algorithm. International Conference in Swarm Intelligence (2011).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property m: int

Number of clusters.

property p_replacement_cluster: float

Probability of replacing a random cluster.

property p_single_cluster: float

Probability of selecting a single cluster.

property p_single_best: float

Probability of selecting the best idea from a single cluster.

property p_double_best: float

Probability of selecting the best idea from a pair of clusters.

property k: float

Controls the sigmoid's slope.

_clusterize(agents: List[Agent]) → Tuple[ndarray, ndarray]

Performs the clusterization over the agents' positions.

Parameters

agents – List of agents.

Returns

Agents indexes and best agent index per cluster.

Return type

(Tuple[np.ndarray, np.ndarray])

_sigmoid(x: float) → float

Calculates the sigmoid function.

Parameters

x – Input value.

Returns

Output value.

update(space: Space, function: Function, iteration: int, n_iterations: int) → None

Wraps Brain Storm Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Number of iterations.s

5.6.2 opytimizer.optimizers.social.ci

Cohort Intelligence.

class opytimizer.optimizers.social.ci.CI(params: Dict[str, Any] | None = None)

A CI class, inherited from Optimizer.

This is the designed class to define CI-related variables and methods.

References

A. J. Kulkarni, I. P. Durugkar, M. Kumar. Cohort Intelligence: A Self Supervised Learning Behavior. IEEE International Conference on Systems, Man, and Cybernetics (2013).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property r: float

Sampling interval reduction factor.

property t: int

Number of variations.

property lower: ndarray

Array of lower bounds.

property upper: ndarray

Array of upper bounds.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space, function: Function*) → None

Wraps Cohort Intelligence over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.6.3 opyoptimizer.optimizers.social.isa

Interactive Search Algorithm.

class opyoptimizer.optimizers.social.isa.**ISA**(*params: Dict[str, Any] | None = None*)

An ISA class, inherited from Optimizer.

This is the designed class to define ISA-related variables and methods.

References

A. Mortazavi, V. Toğan and A. Nuhoğlu. Interactive search algorithm: A new hybrid metaheuristic optimization algorithm. Engineering Applications of Artificial Intelligence (2018).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property w: float

Inertia weight.

property tau: float

Tendency factor.

property local_position: ndarray

Array of velocities.

property velocity: ndarray

Array of velocities.

compile(space: Space) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

evaluate(space: Space, function: Function) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(space: Space, function: Function) → None

Wraps Interactive Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.6.4 opyimizer.optimizers.social.mvpa

Most Valuable Player Algorithm.

class opyimizer.optimizers.social.mvpa.MVPA(params: Dict[str, Any] | None = None)

A MVPA class, inherited from Optimizer.

This is the designed class to define MVPA-related variables and methods.

References

H. Boucekara. Most Valuable Player Algorithm: a novel optimization algorithm inspired from sport. Operational Research (2017).

__init__(params: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_teams: int

Maximum number of teams.

property n_p: int

Number of players per team.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_get_agents_from_team(*agents*: List[[Agent](#)], *index*: int) → List[[Agent](#)]

Gets a set of agents from a specified team.

Parameters

- **agents** – List of agents.
- **index** – Index of team.

Returns

A sorted list of agents that belongs to the specified team.

Return type

(List[[Agent](#)])

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Most Valuable Player Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.6.5 opytimizer.optimizers.social.qsa

Queuing Search Algorithm.

class opytimizer.optimizers.social.qsa.QSA(*params*: Dict[str, Any] | None = None)

A QSA class, inherited from Optimizer.

This is the designed class to define QSA-related variables and methods.

References

J. Zhang et al. Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. Applied Mathematical Modelling (2018).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_calculate_queue(*n_agents*: int, *t_1*: float, *t_2*: float, *t_3*: float) → Tuple[int, int, int]

Calculates the number of agents that belongs to each queue.

Parameters

- **n_agents** – Number of agents.
- **t_1** – Fitness value of first agent in the population.
- **t_2** – Fitness value of second agent in the population.
- **t_3** – Fitness value of third agent in the population.

Returns

The number of agents in first, second and third queues.

Return type

(Tuple[int, int, int])

_business_one(*agents*: List[Agent], *function*: Function, *beta*: float) → None

Performs the first business phase.

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.
- **beta** – Range of fluctuation.

_business_two(*agents*: List[Agent], *function*: Function) → None

Performs the second business phase.

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.

_business_three(*agents*: List[Agent], *function*: Function) → None

Performs the third business phase.

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.

update(*space*: Space, *function*: Function, *iteration*: int, *n_iterations*: int) → None

Wraps Queue Search Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.6.6 opytimizer.optimizers.social.ssd

Social Ski Driver.

class opytimizer.optimizers.social.ssd.SSD(*params*: Dict[str, Any] | None = None)

An SSD class, inherited from Optimizer.

This is the designed class to define SSD-related variables and methods.

References

A. Tharwat and T. Gabel. Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm. Neural Computing and Applications (2019).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c: float

Exploration parameter.

property decay: float

Decay rate.

property local_position: ndarray

Array of local positions.

property velocity: ndarray

Array of velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_mean_global_solution(*alpha: ndarray, beta: ndarray, gamma: ndarray*) → ndarray

Calculates the mean global solution (eq. 9).

Parameters

- **alpha** – 1st agent's current position.
- **beta** – 2nd agent's current position.
- **gamma** – 3rd agent's current position.

Returns

Mean global solution.

Return type

(np.ndarray)

_update_position(*position: ndarray, index: int*) → ndarray

Updates a particle position (eq. 10).

Parameters

- **position** – Agent's current position.
- **index** – Index of current agent.

Returns

A new position.

Return type

(np.ndarray)

_update_velocity(*position: ndarray, mean: ndarray, index: int*) → ndarray

Updates a particle velocity (eq. 11).

Parameters

- **position** – Agent's current position.
- **mean** – Mean global best position.

- **index** – Index of current agent.

Returns

A new velocity.

Return type

(np.ndarray)

evaluate(*space*: [Space](#), *function*: [Function](#)) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Social Ski Driver over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

An evolutionary package for all common opyoptimizer modules. It contains implementations of human social behavior-based optimizers.

5.7 opyoptimizer.optimizers.swarm

5.7.1 opyoptimizer.optimizers.swarm.abc

Artificial Bee Colony.

class opyoptimizer.optimizers.swarm.abc.**ABC**(*params*: Dict[str, Any] | None = None)

An ABC class, inherited from Optimizer.

This is the designed class to define ABC-related variables and methods.

References

D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization (2007).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property **n_trials**: int

Number of trial limits.

property **trial**: ndarray

Array of trial.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

`_evaluate_location`(*agent*: [Agent](#), *neighbour*: [Agent](#), *function*: [Function](#), *index*: *int*) → None

Evaluates a food source location and update its value if possible (eq. 2.2).

Parameters

- **agent** – An agent.
- **neighbour** – A neighbour agent.
- **function** – A function object.
- **index** – Index of trial.

`_send_employee`(*agents*: *List*[[Agent](#)], *function*: [Function](#)) → None

Sends employee bees onto food source to evaluate its nectar.

Parameters

- **agents** – List of agents.
- **function** – A function object.

`_send_onlooker`(*agents*: *List*[[Agent](#)], *function*: [Function](#)) → None

Sends onlooker bees to select new food sources (eq. 2.1).

Parameters

- **agents** – List of agents.
- **function** – A function object.

`_send_scout`(*agents*: *List*[[Agent](#)], *function*: [Function](#)) → None

Sends scout bees to scout for new possible food sources.

Parameters

- **agents** – List of agents.
- **function** – A function object.

`update`(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Artificial Bee Colony over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.2 opytimizer.optimizers.swarm.abo

Artificial Butterfly Optimization.

class opytimizer.optimizers.swarm.abo.**ABO**(*params*: *Dict*[*str*, *Any*] | None = None)

An ABO class, inherited from Optimizer.

This is the designed class to define ABO-related variables and methods.

References

X. Qi, Y. Zhu and H. Zhang. A new meta-heuristic butterfly-inspired algorithm. Journal of Computational Science (2017).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property sunspot_ratio: float

Ratio of sunspot butterflies.

property a: float

Free flight constant.

_flight_mode(*agent: Agent, neighbour: Agent, function: Function*) → Tuple[Agent, bool]

Flies to a new location according to the flight mode (eq. 1).

Parameters

- **agent** – Current agent.
- **neighbour** – Selected neighbour.
- **function** – A Function object that will be used as the objective function.

Returns

Current agent or an agent with updated position, along with a boolean that indicates whether agent is better or not than current one.

Return type

(Tuple[Agent, bool])

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Artificial Butterfly Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.3 opyimizer.optimizers.swarm.af

Artificial Flora.

class opyimizer.optimizers.swarm.af.**AF**(*params: Dict[str, Any] | None = None*)

An AF class, inherited from Optimizer.

This is the designed class to define AF-related variables and methods.

References

L. Cheng, W. Xue-han and Y. Wang. Artificial flora (AF) optimization algorithm. Applied Sciences (2018).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c1: float

First learning coefficient.

property c2: float

Second learning coefficient.

property m: int

Amount of branches.

property Q: float

Selective probability.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Artificial Flora over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.4 opyimizer.optimizers.swarm.ba

Bat Algorithm.

class opyimizer.optimizers.swarm.ba.**BA**(*params*: *Dict[str, Any] | None = None*)

A BA class, inherited from Optimizer.

This is the designed class to define BA-related variables and methods.

References

X.-S. Yang. A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (2010).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property f_min: float

Minimum frequency range.

property f_max: float

Maximum frequency range.

property A: float

Loudness parameter.

property r: float

Pulse rate.

property frequency: ndarray

Array of frequencies.

property velocity: ndarray

Array of velocities.

property loudness: ndarray

Array of loudnesses.

property pulse_rate: ndarray

Array of pulse rates.

compile(*space*: Space) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space*: Space, *function*: Function, *iteration*: int) → None

Wraps Bat Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.

5.7.5 opytimizer.optimizers.swarm.boa

Butterfly Optimization Algorithm.

class opytimizer.optimizers.swarm.boa.BOA(*params*: Dict[str, Any] | None = None)

A BOA class, inherited from Optimizer.

This is the designed class to define BOA-related variables and methods.

References

S. Arora and S. Singh. Butterfly optimization algorithm: a novel approach for global optimization. Soft Computing (2019).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c: float

Sensor modality.

property a: float

Power exponent.

property p: float

Switch probability.

property fragrance: ndarray

Array of fragrances.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_best_movement(*agent_position*: ndarray, *best_position*: ndarray, *fragrance*: ndarray, *random*: float) → ndarray

Updates the agent's position towards the best butterfly (eq. 2).

Parameters

- **agent_positio** – Agent's current position.
- **best_positio** – Best agent's current position.
- **fragrance** – Agent's current fragrance value.
- **random** – A random number between 0 and 1.

Returns

A new position based on best movement.

Return type

(np.ndarray)

_local_movement(*agent_position*: ndarray, *j_position*: ndarray, *k_position*: ndarray, *fragrance*: ndarray, *random*: float) → ndarray

Updates the agent's position using a local movement (eq. 3).

Parameters

- **agent_positio** – Agent's current position.
- **j_positio** – Agent *j* current position.
- **k_positio** – Agent *k* current position.
- **fragrance** – Agent's current fragrance value.
- **random** – A random number between 0 and 1.

Returns

A new position based on local movement.

Return type

(np.ndarray)

update(*space*: [Space](#)) → None

Wraps Butterfly Optimization Algorithm over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.7.6 opyimizer.optimizers.swarm.bwo

Black Widow Optimization.

class opyimizer.optimizers.swarm.bwo.**BWO**(*params*: Dict[str, Any] | None = None)

A BWO class, inherited from Optimizer.

This is the designed class to define BWO-related variables and methods.

References

V. Hayyolalam and A. Kazem. Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence (2020).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property pp: float

Procreating rate.

property cr: float

Cannibalism rate.

property pm: float

Mutation rate.

_procreating(*x1: Agent, x2: Agent*) → Tuple[Agent, Agent]

Procreates a pair of parents into offsprings (eq. 1).

Parameters

- **x1** – Father to produce the offsprings.
- **x2** – Mother to produce the offsprings.

Returns

Two generated offsprings based on parents.

Return type

(Tuple[Agent, Agent])

_mutation(*alpha: Agent*) → Agent

Performs the mutation over an offspring (s. 3.4).

Parameters

alpha – Offspring to be mutated.

Returns

The mutated offspring.

Return type

(Agent)

update(*space: Space, function: Function*) → None

Wraps Black Widow Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.7 opytimizer.optimizers.swarm.cs

Cuckoo Search.

class opytimizer.optimizers.swarm.cs.CS(*params: Dict[str, Any] | None = None*)

A CS class, inherited from Optimizer.

This is the designed class to define CS-related variables and methods.

References

X.-S. Yang and D. Suash. Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (2009).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

Step size.

property beta: float

Lévy distribution parameter.

property p: float

Probability of replacing worst nests.

_generate_new_nests(*agents: List[Agent], best_agent: Agent*) → List[Agent]

Generate new nests (eq. 1).

Parameters

- **agents** – List of agents.
- **best_agent** – Global best agent.

Returns

A new list of agents which can be seen as new nests.

Return type

(List[Agent])

_generate_abandoned_nests(*agents: List[Agent], prob: float*) → List[Agent]

Generate a fraction of nests to be replaced.

Parameters

- **agents** – List of agents.
- **prob** – Probability of replacing worst nests.

Returns

A new list of agents which can be seen as the new nests to be replaced.

Return type

(List[Agent])

_evaluate_nests(*agents: List[Agent], new_agents: List[Agent], function: Function*) → None

Evaluate new nests according to a fitness function.

Parameters

- **agents** – List of current agents.
- **new_agents** – List of new agents to be evaluated.
- **function** – Fitness function used to evaluate.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Cuckoo Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.8 opyimizer.optimizers.swarm.csa

Crow Search Algorithm.

class opyimizer.optimizers.swarm.csa.CSA(*params*: *Dict[str, Any]* | *None* = *None*)

A CSA class, inherited from Optimizer.

This is the designed class to define CSA-related variables and methods.

References

A. Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures (2016).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property fl: float

Flight length.

property AP: float

Awareness probability.

property memory: ndarray

Array of memories.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

evaluate(*space*: [Space](#), *function*: [Function](#)) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(*space*: [Space](#)) → None

Wraps Crow Search Algorithm over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.7.9 opyoptimizer.optimizers.swarm.eho

Elephant Herding Optimization.

class opyoptimizer.optimizers.swarm.eho.**EHO**(*params: Dict[str, Any] | None = None*)

An EHO class, inherited from Optimizer.

This is the designed class to define EHO-related variables and methods.

References

G.-G. Wang, S. Deb and L. Coelho. Elephant Herding Optimization. International Symposium on Computational and Business Intelligence (2015).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

Matriarch influence.

property beta: float

Center influence.

property n_clans: int

Maximum number of clans.

property n_ci: int

Number of elephants per clan.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_get_agents_from_clan(*agents: List[Agent], index: int*) → List[Agent]

Gets a set of agents from a specified clan.

Parameters

- **agents** – List of agents.
- **index** – Index of clan.

Returns

A sorted list of agents that belongs to the specified clan.

Return type

(List[Agent])

_updating_operator(*agents: List[Agent], centers: ndarray, function: Function*) → None

Performs the separating operator.

Parameters

- **agents** – List of agents.
- **centers** – List of centers.
- **function** – A Function object that will be used as the objective function.

_separating_operator(*agents*: List[Agent]) → None

Performs the separating operator.

Parameters

agents – List of agents.

update(*space*: Space, *function*: Function) → None

Wraps Elephant Herd Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.10 opytimizer.optimizers.swarm.fa

Firefly Algorithm.

class opytimizer.optimizers.swarm.fa.FA(*params*: Dict[str, Any] | None = None)

A FA class, inherited from Optimizer.

This is the designed class to define FA-related variables and methods.

References

X.-S. Yang. Firefly algorithms for multimodal optimization. International symposium on stochastic algorithms (2009).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property alpha: float

Randomization parameter.

property beta: float

Attractiveness parameter.

property gamma: float

Light absorption coefficient.

update(*space*: Space, *n_iterations*: int) → None

Wraps Firefly Algorithm over all agents and variables (eq. 3-9).

Parameters

- **space** – Space containing agents and update-related information.
- **n_iterations** – Maximum number of iterations.

5.7.11 opytimizer.optimizers.swarm.ffoa

Fruit-Fly Optimization Algorithm.

class opytimizer.optimizers.swarm.ffoa.FFOA(*params*: Dict[str, Any] | None = None)

A FFOA class, inherited from Optimizer.

This is the designed class to define FFOA-related variables and methods.

References

W.-T. Pan. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems (2012).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property x_axis: List[Agent]

x axis.

property y_axis: List[Agent]

y axis.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space, function: Function*) → None

Wraps Fruit-Fly Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.12 opytimizer.optimizers.swarm.fpa

Flower Pollination Algorithm.

class opytimizer.optimizers.swarm.fpa.FPA(*params: Dict[str, Any] | None = None*)

A FPA class, inherited from Optimizer.

This is the designed class to define FPA-related variables and methods.

References

X.-S. Yang. Flower pollination algorithm for global optimization. International conference on unconventional computing and natural computation (2012).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property beta: float

Lévy flight control parameter.

property eta: float

Lévy flight scaling factor.

property p: float

Probability of local pollination.

_global_pollination(*agent_position: ndarray, best_position: ndarray*) → ndarray

Updates the agent's position based on a global pollination (eq. 1).

Parameters

- **agent_position** – Agent's current position.
- **best_position** – Best agent's current position.

Returns

A new position.

Return type

(np.ndarray)

_local_pollination(*agent_position: ndarray, k_position: ndarray, l_position: ndarray, epsilon: float*) → ndarray

Updates the agent's position based on a local pollination (eq. 3).

Parameters

- **agent_position** – Agent's current position.
- **k_position** – Agent's (index k) current position.
- **l_position** – Agent's (index l) current position.
- **epsilon** – An uniform random generated number.

Returns

A new position.

Return type

(np.ndarray)

update(*space: Space, function: Function*) → None

Wraps Flower Pollination Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.13 opyimizer.optimizers.swarm.fso

Flying Squirrel Optimizer.

class opyimizer.optimizers.swarm.fso.FSO(*params: Dict[str, Any] | None = None*)

A FSO class, inherited from Optimizer.

This is the designed class to define FSO-related variables and methods.

References

G. Azizyan et al. Flying Squirrel Optimizer (FSO): A novel SI-based optimization algorithm for engineering problems. Iranian Journal of Optimization (2019).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property beta: float

Lévy distribution parameter.

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Flying Squirrel Optimizer over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.14 opyimizer.optimizers.swarm.goa

Grasshopper Optimization Algorithm.

class opyimizer.optimizers.swarm.goa.GOA(*params: Dict[str, Any] | None = None*)

A GOA class, inherited from Optimizer.

This is the designed class to define GOA-related variables and methods.

References

S. Saremi, S. Mirjalili and A. Lewis. Grasshopper Optimisation Algorithm: Theory and application. Advances in Engineering Software (2017).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property c_min: float

Minimum comfort zone.

property c_max: float

Maximum comfort zone.

property f: float

Intensity of attraction.

property l: float

Attractive length scale.

_social_force(*r: ndarray*) → ndarray

Calculates the social force based on an input value.

Parameters

r – Array of values.

Returns

The social force based on the input value.

Return type

(np.ndarray)

update(*space*: [Space](#), *function*: [Function](#), *iteration*: *int*, *n_iterations*: *int*) → *None*

Wraps Grasshopper Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.15 opyoptimizer.optimizers.swarm.js

Jellyfish Search-based algorithms.

class `opyoptimizer.optimizers.swarm.js.JS`(*params*: *Dict[str, Any]* | *None = None*)

A JS class, inherited from `Optimizer`.

This is the designed class to define JS-related variables and methods.

References

J.-S. Chou and D.-N. Truong. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation* (2020).

__init__(*params*: *Dict[str, Any]* | *None = None*) → *None*

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property eta: *float*

Chaotic map coefficient.

property beta: *float*

Distribution coefficient.

property gamma: *float*

Motion coefficient.

_initialize_chaotic_map(*agents*: *List[Agent]*) → *None*

Initializes a set of agents using a logistic chaotic map.

Parameters

agents – List of agents.

compile(*space*: [Space](#)) → *None*

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_ocean_current(*agents*: *List[Agent]*, *best_agent*: [Agent](#)) → *ndarray*

Calculates the ocean current (eq. 9).

Parameters

- **agents** – List of agents.
- **best_agent** – Best agent.

Returns

A trend value for the ocean current.

Return type

(np.ndarray)

_motion_a(*lb: ndarray, ub: ndarray*) → ndarray

Calculates type A motion (eq. 12).

Parameters

- **lb** – Array of lower bounds.
- **ub** – Array of upper bounds.

Returns

A type A motion array.

Return type

(np.ndarray)

_motion_b(*agent_i: Agent, agent_j: Agent*) → ndarray

Calculates type B motion (eq. 15).

Parameters

- **agent_i** – Current agent to be updated.
- **agent_j** – Selected agent.

Returns

A type B motion array.

Return type

(np.ndarray)

update(*space: Space, iteration: int, n_iterations: int*) → None

Wraps Jellyfish Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

class opyimizer.optimizers.swarm.js.NBJS(*params: Dict[str, Any] | None = None*)

An NBJS class, inherited from JS.

This is the designed class to define NBJS-related variables and methods.

References

Publication pending.

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_motion_a(*lb: ndarray, ub: ndarray*) → ndarray

Calculates type A motion.

Parameters

- **lb** – Array of lower bounds.
- **ub** – Array of upper bounds.

Returns

A type A motion array.

Return type

(np.ndarray)

5.7.16 opyimizer.optimizers.swarm.kh

Krill Herd.

class opyimizer.optimizers.swarm.kh.**KH**(*params: Dict[str, Any] | None = None*)

A KH class, inherited from Optimizer.

This is the designed class to define KH-related variables and methods.

References

A. Gandomi and A. Alavi. Krill herd: A new bio-inspired optimization algorithm. Communications in Nonlinear Science and Numerical Simulation (2012).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property N_max: float

Maximum induced speed.

property w_n: float

Inertia weight of the neighbours' motion.

property NN: int

Number of neighbours.

property V_f: float

Foraging speed.

property w_f: float

Inertia weight of the foraging motion.

property D_max: float

Maximum diffusion speed.

property C_t: float

Position constant.

property Cr: float

Crossover probability.

property Mu: float

Mutation probability.

property motion: ndarray

Array of motions.

property foraging: ndarray

Array of foragings.

compile(space: Space) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_food_location(agents: List[Agent], function: Function) → Agent

Calculates the food location.

Parameters

- **agents** – List of agents.
- **function** – A Function object that will be used as the objective function.

Returns

A new food location.

Return type

(Agent)

_sensing_distance(agents: List[Agent], idx: int) → Tuple[float, float]

Calculates the sensing distance for an individual krill (eq. 7).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.

Returns

The sensing distance for an individual krill.

Return type

(Tuple[float, float])

_get_neighbours(agents: List[Agent], idx: int, sensing_distance: float, eucl_distance: List[float]) → List[Agent]

Gathers the neighbours based on the sensing distance.

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.
- **sensing_distance** – Sensing distanced used to gather the krill's neighbours.
- **eucl_distance** – List of euclidean distances.

Returns

A list containing the krill's neighbours.

Return type

(List[Agent])

_local_alpha(*agent*: [Agent](#), *worst*: [Agent](#), *best*: [Agent](#), *neighbours*: *List*[[Agent](#)]) → float

Calculates the local alpha (eq. 4).

Parameters

- **agent** – Selected agent.
- **worst** – Worst agent.
- **best** – Best agent.
- **neighbours** – List of neighbours.

Returns

The local alpha.

Return type

(float)

_target_alpha(*agent*: [Agent](#), *worst*: [Agent](#), *best*: [Agent](#), *C_best*: float) → float

Calculates the target alpha (eq. 8).

Parameters

- **agent** – Selected agent.
- **worst** – Worst agent.
- **best** – Best agent.
- **C_best** – Effectiveness coefficient.

Returns

The target alpha.

Return type

(float)

_neighbour_motion(*agents*: *List*[[Agent](#)], *idx*: int, *iteration*: int, *n_iterations*: int, *motion*: ndarray) → ndarray

Performs the motion induced by other krill individuals (eq. 2).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.
- **motion** – Array of motions.

Returns

The krill's neighbour motion.

Return type

(np.ndarray)

_food_beta(*agent*: [Agent](#), *worst*: [Agent](#), *best*: [Agent](#), *food*: ndarray, *C_food*: float) → ndarray

Calculates the food attraction (eq. 13).

Parameters

- **agent** – Selected agent.

- **worst** – Worst agent.
- **best** – Best agent.
- **food** – Food location.
- **C_food** – Food coefficient.

Returns

The food attraction.

Return type

(np.ndarray)

_best_beta(*agent*: Agent, *worst*: Agent, *best*: Agent) → ndarray

Calculates the best attraction (eq. 15).

Parameters

- **agent** – Selected agent.
- **worst** – Worst agent.
- **best** – Best agent.

Returns

The best attraction.

Return type

(np.ndarray)

_foraging_motion(*agents*: List[Agent], *idx*: int, *iteration*: int, *n_iterations*: int, *food*: ndarray, *foraging*: ndarray) → ndarray

Performs the foraging induced by the food location (eq. 10).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.
- **food** – Food location.
- **foraging** – Array of foraging motions.

Returns

The krill's foraging motion.

Return type

(np.ndarray)

_physical_diffusion(*n_variables*: int, *n_dimensions*: int, *iteration*: int, *n_iterations*: int) → float

Performs the physical diffusion of individual krills (eq. 16-17).

Parameters

- **n_variables** – Number of decision variables.
- **n_dimensions** – Number of dimensions.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

The physical diffusion.

Return type

(float)

_update_position(*agents*: List[Agent], *idx*: int, *iteration*: int, *n_iterations*: int, *food*: ndarray, *motion*: ndarray, *foraging*: ndarray) → ndarray

Updates a single krill position (eq. 18-19).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.
- **food** – Food location.
- **motion** – Array of motions.
- **foraging** – Array of foraging motions.

Returns

The updated position.

Return type

(np.ndarray)

_crossover(*agents*: List[Agent], *idx*: int) → Agent

Performs the crossover between selected agent and a randomly agent (eq. 21).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.

Returns

An agent after suffering a crossover operator.

Return type

(Agent)

_mutation(*agents*: List[Agent], *idx*: int) → Agent

Performs the mutation between selected agent and randomly agents (eq. 22).

Parameters

- **agents** – List of agents.
- **idx** – Selected agent.

Returns

An agent after suffering a mutation operator.

Return type

(Agent)

update(*space*: Space, *function*: Function, *iteration*: int, *n_iterations*: int) → None

Wraps motion and genetic updates over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.17 opyoptimizer.optimizers.swarm.mfo

Moth-Flame Optimization.

class opyoptimizer.optimizers.swarm.mfo.**MFO**(*params: Dict[str, Any] | None = None*)

A MFO class, inherited from Optimizer.

This is the designed class to define MFO-related variables and methods.

References

S. Mirjalili. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems (2015).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property b: float

Spiral constant.

update(*space: Space, iteration: int, n_iterations: int*) → None

Wraps Moth-Flame Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.18 opyoptimizer.optimizers.swarm.mrfo

Manta Ray Foraging Optimization.

class opyoptimizer.optimizers.swarm.mrfo.**MRFO**(*params: Dict[str, Any] | None = None*)

An MRFO class, inherited from Optimizer.

This is the designed class to define MRFO-related variables and methods.

References

W. Zhao, Z. Zhang and L. Wang. Manta Ray Foraging Optimization: An effective bio-inspired optimizer for engineering applications. Engineering Applications of Artificial Intelligence (2020).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property S: float

Somersault foraging.

_cyclone_foraging(*agents: List[Agent], best_position: ndarray, i: int, iteration: int, n_iterations: int*) → ndarray

Performs the cyclone foraging procedure (eq. 3-7).

Parameters

- **agents** – List of agents.
- **best_position** – Global best position.
- **i** – Index of current manta ray.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

Returns

A new cyclone foraging.

Return type

(np.ndarray)

_chain_foraging(*agents: List[Agent], best_position: ndarray, i: int*) → ndarray

Performs the chain foraging procedure (eq. 1-2).

Parameters

- **agents** – List of agents.
- **best_position** – Global best position.
- **i** – Index of current manta ray.

Returns

A new chain foraging.

Return type

(np.ndarray)

_somersault_foraging(*position: ndarray, best_position: ndarray*) → ndarray

Performs the somersault foraging procedure (eq. 8).

Parameters

- **position** – Agent's current position.
- **best_position** – Global best position.

Returns

A new somersault foraging.

Return type

(np.ndarray)

update(*space: Space, function: Function, iteration: int, n_iterations: int*) → None

Wraps Manta Ray Foraging Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.19 opyimizer.optimizers.swarm.pio

Pigeon-Inspired Optimization.

class opyimizer.optimizers.swarm.pio.PIO(*params: Dict[str, Any] | None = None*)

A PIO class, inherited from Optimizer.

This is the designed class to define PIO-related variables and methods.

References

H. Duan and P. Qiao. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. International Journal of Intelligent Computing and Cybernetics (2014).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property n_c1: int

Number of mapping iterations.

property n_c2: int

Number of landmark iterations.

property R: float

Map and compass factor.

property n_p: int

Number of pigeons.

property velocity: ndarray

Array of pulse rates.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_calculate_center(*agents: List[Agent]*) → ndarray

Calculates the center position (eq. 8).

Parameters

agents – List of agents.

Returns

The center position.

Return type

(np.ndarray)

_update_center_position(*position: ndarray, center: ndarray*) → None

Updates a pigeon position based on the center (eq. 9).

Parameters

- **position** – Agent’s current position.
- **center** – Center position.

Returns

A new center-based position.

Return type

(np.ndarray)

update(*space*: [Space](#), *iteration*: int) → None

Wraps Pigeon-Inspired Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.

5.7.20 opyoptimizer.optimizers.swarm.pso

Particle Swarm Optimization-based algorithms.

class opyoptimizer.optimizers.swarm.pso.**PSO**(*params*: Dict[str, Any] | None = None)

A PSO class, inherited from Optimizer.

This is the designed class to define PSO-related variables and methods.

References

J. Kennedy, R. C. Eberhart and Y. Shi. Swarm intelligence. Artificial Intelligence (2001).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property w: float

Inertia weight.

property c1: float

Cognitive constant.

property c2: float

Social constant.

property local_position: ndarray

Array of velocities.

property velocity: ndarray

Array of velocities.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

evaluate(*space*: [Space](#), *function*: [Function](#)) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(*space*: [Space](#)) → None

Wraps Particle Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

class `opyoptimizer.optimizers.swarm.pso.AIWPSO`(*params*: *Dict[str, Any] | None = None*)

An AIWPSO class, inherited from PSO.

This is the designed class to define AIWPSO-related variables and methods.

References

A. Nickabadi, M. M. Ebadzadeh and R. Safabakhsh. A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing* (2011).

__init__(*params*: *Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property w_min: float

Minimum inertia weight.

property w_max: float

Maximum inertia weight.

property fitness: List[float]

List of fitnesses.

_compute_success(*agents*: List[[Agent](#)]) → None

Computes the particles' success for updating inertia weight (eq. 16).

Parameters

agents – List of agents.

update(*space*: [Space](#), *iteration*: int) → None

Wraps Adaptive Inertia Weight Particle Swarm Optimization over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.

class `opyoptimizer.optimizers.swarm.pso.RPSO`(*params*: *Dict[str, Any] | None = None*)

An RPSO class, inherited from Optimizer.

This is the designed class to define RPSO-related variables and methods.

References

M. Roder, G. H. de Rosa, L. A. Passos, A. L. D. Rossi and J. P. Papa. Harnessing Particle Swarm Optimization Through Relativistic Velocity. IEEE Congress on Evolutionary Computation (2020).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property mass: ndarray

Array of masses.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space*) → None

Wraps Relativistic Particle Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

class opyimizer.optimizers.swarm.pso.**SAVPSO**(*params: Dict[str, Any] | None = None*)

An SAVPSO class, inherited from Optimizer.

This is the designed class to define SAVPSO-related variables and methods.

References

H. Lu and W. Chen. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. Journal of global optimization (2008).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

update(*space: Space*) → None

Wraps Self-adaptive Velocity Particle Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

class opyimizer.optimizers.swarm.pso.**VPSO**(*params: Dict[str, Any] | None = None*)

A VPSO class, inherited from Optimizer.

This is the designed class to define VPSO-related variables and methods.

References

W.-P. Yang. Vertical particle swarm optimization algorithm and its application in soft-sensor modeling. International Conference on Machine Learning and Cybernetics (2007).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property v_velocity: ndarray

Array of vertical velocities.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space: Space*) → None

Wraps Vertical Particle Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.7.21 opyoptimizer.optimizers.swarm.sbo

Satin Bowerbird Optimizer.

class opyoptimizer.optimizers.swarm.sbo.**SBO**(*params: Dict[str, Any] | None = None*)

A SBO class, inherited from Optimizer.

This is the designed class to define SBO-related variables and methods.

References

S. H. S. Moosavi and V. K. Bardsiri. Satin bowerbird optimizer: a new optimization algorithm to optimize ANFIS for software development effort estimation. Engineering Applications of Artificial Intelligence (2017).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the mp_mutation-heuristics.

property alpha: float

Step size.

property p_mutation: float

Probability of mutation.

property z: float

Percentage of width between lower and upper bounds.

property sigma: List[float]

List of widths.

compile(*space: Space*) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Satin Bowerbird Optimizer over all agents and variables (eq. 1-7).

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.22 opyoptimizer.optimizers.swarm.sca

Sine Cosine Algorithm.

class opyoptimizer.optimizers.swarm.sca.SCA(*params*: *Dict[str, Any]* | None = None)

A SCA class, inherited from Optimizer.

This is the designed class to define SCA-related variables and methods.

References

S. Mirjalili. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems (2016).

__init__(*params*: *Dict[str, Any]* | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property r_min: float

Minimum function range.

property r_max: float

Maximum function range.

property a: float

Loudness parameter.

_update_position(*agent_position*: ndarray, *best_position*: ndarray, *r1*: float, *r2*: float, *r3*: float, *r4*: float) → ndarray

Updates a single particle position over a single variable (eq. 3.3).

Parameters

- **agent_position** – Agent’s current position.
- **best_position** – Global best position.
- **r1** – Controls the next position’s region.
- **r2** – Defines how far the movement should be.
- **r3** – Random weight for emphasizing or deemphasizing the movement.
- **r4** – Random number to decide whether sine or cosine should be used.

Returns

A new position.

Return type

(np.ndarray)

update(*space*: [Space](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Sine Cosine Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.23 opyoptimizer.optimizers.swarm.sfo

Sailfish Optimizer.

class opyoptimizer.optimizers.swarm.sfo.**SFO**(*params*: *Dict[str, Any]* | *None* = *None*)

A SFO class, inherited from Optimizer.

This is the designed class to define SFO-related variables and methods.

References

S. Shadravan, H. Naji and V. Bardsiri. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Engineering Applications of Artificial Intelligence (2019).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property PP: float

Percentage of initial sailfishes.

property A: int

Attack power coefficient.

property e: float

Attack power decrease.

property sardines: List[[Agent](#)]

List of sardines.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

_generate_random_agent(*agent*: [Agent](#)) → [Agent](#)

Generates a new random-based agent.

Parameters

agent – Agent to be copied.

Returns

Random-based agent.

Return type

([Agent](#))

_calculate_lambda_i(*n_sailfishes*: int, *n_sardines*: int) → float

Calculates the lambda value (eq. 7).

Parameters

- **n_sailfishes** (int) – Number of sailfishes.
- **n_sardines** (int) – Number of sardines.

Returns

Lambda value from current iteration.

Return type

(float)

_update_sailfish(*agent*: Agent, *best_agent*: Agent, *best_sardine*: Agent, *lambda_i*: float) → ndarray

Updates the sailfish's position (eq. 6).

Parameters

- **agent** – Current agent's.
- **best_agent** – Best sailfish.
- **best_sardine** – Best sardine.
- **lambda_i** – Lambda value.

Returns

An updated position.

Return type

(np.ndarray)

update(*space*: Space, *function*: Function, *iteration*: int) → None

Wraps Sailfish Optimizer over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.
- **iteration** – Current iteration.

5.7.24 opytimizer.optimizers.swarm.sos

Symbiotic Organisms Search.

class opytimizer.optimizers.swarm.sos.SOS(*params*: Dict[str, Any] | None = None)

An SOS class, inherited from Optimizer.

This is the designed class to define SOS-related variables and methods.

References

M.-Y. Cheng and D. Prayogo. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Computers & Structures (2014).

__init__(*params*: Dict[str, Any] | None = None) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

_mutualism(*agent_i*: [Agent](#), *agent_j*: [Agent](#), *best_agent*: [Agent](#), *function*: [Function](#)) → None

Performs the mutualism operation.

Parameters

- **agent_i** – Selected *i* agent.
- **agent_j** – Selected *j* agent.
- **best_agent** – Global best agent.
- **function** – A Function object that will be used as the objective function.

_commensalism(*agent_i*: [Agent](#), *agent_j*: [Agent](#), *best_agent*: [Agent](#), *function*: [Function](#)) → None

Performs the commensalism operation.

Parameters

- **agent_i** – Selected *i* agent.
- **agent_j** – Selected *j* agent.
- **best_agent** – Global best agent.
- **function** – A Function object that will be used as the objective function.

_parasitism(*agent_i*: [Agent](#), *agent_j*: [Agent](#), *function*: [Function](#)) → None

Performs the parasitism operation.

Parameters

- **agent_i** – Selected *i* agent.
- **agent_j** – Selected *j* agent.
- **function** – A Function object that will be used as the objective function.

update(*space*: [Space](#), *function*: [Function](#)) → None

Wraps Symbiotic Organisms Search over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **function** – A Function object that will be used as the objective function.

5.7.25 opyimizer.optimizers.swarm.ssa

Salp Swarm Algorithm.

class opyimizer.optimizers.swarm.ssa.SSA(*params*: *Dict[str, Any]* | *None* = *None*)

A SSA class, inherited from Optimizer.

This is the designed class to define SSA-related variables and methods.

References

S. Mirjalili et al. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software* (2017).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

- **params** – Contains key-value parameters to the meta-heuristics.

update(*space*: [Space](#), *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Salp Swarm Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.26 opyimizer.optimizers.swarm.sso

Simplified Swarm Optimization.

class opyimizer.optimizers.swarm.sso.SSO(*params*: *Dict[str, Any]* | *None* = *None*)

A SSO class, inherited from Optimizer.

This is the designed class to define SSO-related variables and methods.

References

C. Bae et al. A new simplified swarm optimization (SSO) using exchange local search scheme. International Journal of Innovative Computing, Information and Control (2012).

__init__(*params*: *Dict[str, Any]* | *None* = *None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property C_w: float

Weighing constant.

property C_p: float

Local constant.

property C_g: float

Global constant.

property local_position: ndarray

Array of local positions.

compile(*space*: [Space](#)) → None

Compiles additional information that is used by this optimizer.

Parameters

space – A Space object containing meta-information.

evaluate(*space*: [Space](#), *function*: [Function](#)) → None

Evaluates the search space according to the objective function.

Parameters

- **space** – A Space object that will be evaluated.
- **function** – A Function object that will be used as the objective function.

update(*space*: [Space](#)) → None

Wraps Simplified Swarm Optimization over all agents and variables.

Parameters

space – Space containing agents and update-related information.

5.7.27 opytimizer.optimizers.swarm.stoa

Sooty Tern Optimization Algorithm.

class opytimizer.optimizers.swarm.stoa.STOA(*params: Dict[str, Any] | None = None*)

An STOA class, inherited from Optimizer.

This is the designed class to define STOA-related variables and methods.

References

G. Dhiman and A. Kaur. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Engineering Applications of Artificial Intelligence (2019).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property Cf: float

Controlling variable.

property u: float

Spiral shape first constant.

property v: float

Spiral shape second constant.

update(*space: Space, iteration: int, n_iterations: int*) → None

Wraps Sooty Tern Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** – Maximum number of iterations.

5.7.28 opytimizer.optimizers.swarm.woa

Whale Optimization Algorithm.

class opytimizer.optimizers.swarm.woa.WOA(*params: Dict[str, Any] | None = None*)

A WOA class, inherited from Optimizer.

This is the designed class to define WOA-related variables and methods.

References

S. Mirjalli and A. Lewis. The Whale Optimization Algorithm. Advances in Engineering Software (2016).

__init__(*params: Dict[str, Any] | None = None*) → None

Initialization method.

Parameters

params – Contains key-value parameters to the meta-heuristics.

property b: float

Logarithmic spiral.

_generate_random_agent(*agent*: *Agent*) → *Agent*

Generates a new random-based agent.

Parameters

agent – Agent to be copied.

Returns

Random-based agent.

Return type

(*Agent*)

update(*space*: *Space*, *iteration*: *int*, *n_iterations*: *int*) → None

Wraps Whale Optimization Algorithm over all agents and variables.

Parameters

- **space** – Space containing agents and update-related information.
- **iteration** – Current iteration.
- **n_iterations** (*int*) – Maximum number of iterations

An evolutionary package for all common opyimizer modules. It contains implementations of swarm-based optimizers.

An optimizers package for all common opyimizer modules. It contains specific packages of every optimization taxonomy covered by opyimizer.

OPYOPTIMIZER.SPACES

One can see the space as the place that agents will update their positions and evaluate a fitness function. However, the newest approaches may consider a different type of space. Thinking about that, we are glad to support diverse space implementations.

6.1 opyoptimizer.spaces.boolean

Boolean-based search space.

```
class opyoptimizer.spaces.boolean.BooleanSpace(n_agents: int, n_variables: int, mapping: List[str] | None = None)
```

A BooleanSpace class for agents, variables and methods related to the boolean search space.

```
__init__(n_agents: int, n_variables: int, mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_agents** – Number of agents.
- **n_variables** – Number of decision variables.
- **mapping** – String-based identifiers for mapping variables' names.

```
_initialize_agents() → None
```

Initializes agents with their positions and defines a best agent.

6.2 opyoptimizer.spaces.graph

Graph-based search space.

6.3 opyoptimizer.spaces.grid

Grid-based search space.

```
class opyoptimizer.spaces.grid.GridSpace(n_variables: int, step: float | List | Tuple | ndarray, lower_bound: float | List | Tuple | ndarray, upper_bound: float | List | Tuple | ndarray, mapping: List[str] | None = None)
```

A GridSpace class for agents, variables and methods related to the grid search space.

```
__init__(n_variables: int, step: float | List | Tuple | ndarray, lower_bound: float | List | Tuple | ndarray, upper_bound: float | List | Tuple | ndarray, mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_variables** – Number of decision variables.
- **step** – Variables' steps.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **mapping** – String-based identifiers for mapping variables' names.

property step: ndarray

Step size of each variable.

property grid: ndarray

Grid with possible search values.

_create_grid() → None

Creates a grid of possible search values.

_initialize_agents() → None

Initializes agents with their positions and defines a best agent.

6.4 opyoptimizer.spaces.hyper_complex

Hypercomplex-based search space.

```
class opyoptimizer.spaces.hyper_complex.HyperComplexSpace(n_agents: int, n_variables: int,  
                                                         n_dimensions: int, mapping: List[str] |  
                                                         None = None)
```

An HyperComplexSpace class that will hold agents, variables and methods related to the hypercomplex search space.

```
__init__(n_agents: int, n_variables: int, n_dimensions: int, mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_agents** – Number of agents.
- **n_variables** – Number of decision variables.
- **n_dimensions** – Number of search space dimensions.
- **mapping** – String-based identifiers for mapping variables' names.

_initialize_agents() → None

Initializes agents with their positions and defines a best agent.

6.5 opyoptimizer.spaces.pareto

Pareto-based search space.

```
class opyoptimizer.spaces.pareto.ParetoSpace(data_points: ndarray, mapping: List[str] | None = None)
```

A ParetoSpace class for agents, variables and methods related to the pareto-frontier search space.

__init__(*data_points: ndarray, mapping: List[str] | None = None*) → None

Initialization method.

Parameters

- **data_points** – Pre-defined data points.
- **mapping** – String-based identifiers for mapping variables' names.

_load_agents(*data_points: ndarray*) → None

Loads agents from pre-defined data points.

Parameters

data_points – Pre-defined data points.

build(*data_points: ndarray*) → None

Builds the object by creating and pre-loading the agents.

Parameters

data_points – Pre-defined data points.

clip_by_bound() → None

Overrides default function as no clipping should be performed.

6.6 opytimizer.spaces.search

Traditional-based search space.

class opytimizer.spaces.search.**SearchSpace**(*n_agents: int, n_variables: int, lower_bound: float | List | Tuple | ndarray, upper_bound: float | List | Tuple | ndarray, mapping: List[str] | None = None*)

A SearchSpace class for agents, variables and methods related to the search space.

__init__(*n_agents: int, n_variables: int, lower_bound: float | List | Tuple | ndarray, upper_bound: float | List | Tuple | ndarray, mapping: List[str] | None = None*) → None

Initialization method.

Parameters

- **n_agents** – Number of agents.
- **n_variables** – Number of decision variables.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **mapping** – String-based identifiers for mapping variables' names.

_initialize_agents() → None

Initializes agents with their positions and defines a best agent.

6.7 opytimizer.spaces.tree

Tree-based search space.

class opytimizer.spaces.tree.**TreeSpace**(*n_agents: int, n_variables: int, lower_bound: float | List | Tuple | ndarray, upper_bound: float | List | Tuple | ndarray, n_terminals: int | None = 1, min_depth: int | None = 1, max_depth: int | None = 3, functions: List[str] | None = None, mapping: List[str] | None = None*)

A TreeSpace class for trees, agents, variables and methods related to a tree-based search space.

```
__init__(n_agents: int, n_variables: int, lower_bound: float | List | Tuple | ndarray, upper_bound: float |  
List | Tuple | ndarray, n_terminals: int | None = 1, min_depth: int | None = 1, max_depth: int |  
None = 3, functions: List[str] | None = None, mapping: List[str] | None = None) → None
```

Initialization method.

Parameters

- **n_agents** – Number of agents (trees).
- **n_variables** – Number of decision variables.
- **lower_bound** – Minimum possible values.
- **upper_bound** – Maximum possible values.
- **n_terminals** – Number of terminal nodes.
- **min_depth** – Minimum depth of the trees.
- **max_depth** – Maximum depth of the trees.
- **functions** – Function nodes.
- **mapping** – String-based identifiers for mapping variables' names.

```
property n_terminals: int
```

Number of terminal nodes.

```
property min_depth: int
```

Minimum depth of the trees.

```
property max_depth: int
```

Maximum depth of the trees.

```
property functions: List[str]
```

Function nodes.

```
property terminals: List[str]
```

Terminals nodes.

```
property trees: List[Node]
```

Trees (derived from the Node class).

```
property best_tree: Node
```

Best tree.

```
_create_terminals() → None
```

Creates a list of terminals.

```
_create_trees() → None
```

Creates a list of trees based on the GROW algorithm.

```
_initialize_agents() → None
```

Initializes agents with their positions and defines a best agent.

```
_initialize_terminals() → None
```

Initializes terminals with their positions.

```
grow(min_depth: int | None = 1, max_depth: int | None = 3) → Node
```

Creates a random tree based on the GROW algorithm.

References

S. Luke. Two Fast Tree-Creation Algorithms for Genetic Programming. IEEE Transactions on Evolutionary Computation (2000).

Parameters

- **min_depth** – Minimum depth of the tree.
- **max_depth** – Maximum depth of the tree.

Returns

Random tree based on the GROW algorithm.

Return type

(*Node*)

Customizable space module that provides different search spaces implementations.

OPYTIMIZER.UTILS

This is a utility package. Common things shared across the application should be implemented here. It is better to implement once and use as you wish than re-implementing the same thing repeatedly.

7.1 opytimizer.utils.callback

Callbacks.

class opytimizer.utils.callback.Callback

A Callback class that handles additional variables and methods manipulation that are not provided by the library.

__init__()

Initialization method.

on_task_begin(*opt_model: Opytimizer*) → None

Performs a callback whenever a task begins.

Parameters

opt_model – An instance of the optimization model.

on_task_end(*opt_model: Opytimizer*) → None

Performs a callback whenever a task ends.

Parameters

opt_model – An instance of the optimization model.

on_iteration_begin(*iteration: int, opt_model: Opytimizer*) → None

Performs a callback whenever an iteration begins.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

on_iteration_end(*iteration: int, opt_model: Opytimizer*) → None

Performs a callback whenever an iteration ends.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

on_evaluate_before(**evaluate_args*) → None

Performs a callback prior to the *evaluate* method.

on_evaluate_after(**evaluate_args*) → None

Performs a callback after the *evaluate* method.

on_update_before(**update_args*) → None

Performs a callback prior to the *update* method.

on_update_after(**update_args*) → None

Performs a callback after the *update* method.

class `opyimizer.utils.callback.CallbackVessel`(*callbacks*: List[[Callback](#)])

Wraps multiple callbacks in an ready-to-use class.

__init__(*callbacks*: List[[Callback](#)]) → None

Initialization method.

Parameters

callbacks – List of Callback-based childs.

property `callbacks`: List[[Callback](#)]

List of Callback-based childs.

on_task_begin(*opt_model*: *Opyimizer*) → None

Performs a list of callbacks whenever a task begins.

Parameters

opt_model – An instance of the optimization model.

on_task_end(*opt_model*: *Opyimizer*) → None

Performs a list of callbacks whenever a task ends.

Parameters

opt_model – An instance of the optimization model.

on_iteration_begin(*iteration*: int, *opt_model*: *Opyimizer*) → None

Performs a list of callbacks whenever an iteration begins.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

on_iteration_end(*iteration*: int, *opt_model*: *Opyimizer*) → None

Performs a list of callbacks whenever an iteration ends.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

on_evaluate_before(**evaluate_args*) → None

Performs a list of callbacks prior to the *evaluate* method.

on_evaluate_after(**evaluate_args*) → None

Performs a list of callbacks after the *evaluate* method.

on_update_before(**update_args*) → None

Performs a list of callbacks prior to the *update* method.

on_update_after(**update_args*) → None

Performs a list of callbacks after the *update* method.

class opyimizer.utils.callback.**CheckpointCallback**(*file_path*: str | None = None, *frequency*: int | None = 0)

A CheckpointCallback class that handles additional logging and model's checkpointing.

__init__(*file_path*: str | None = None, *frequency*: int | None = 0) → None

Initialization method.

Parameters

- **file_path** – Path of file to be saved.
- **frequency** – Interval between checkpoints.

property file_path: str

File's path.

property frequency: int

Interval between checkpoints.

on_iteration_end(*iteration*: int, *opt_model*: Opyimizer) → None

Performs a callback whenever an iteration ends.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

class opyimizer.utils.callback.**DiscreteSearchCallback**(*allowed_values*: List[int | float] = None)

A DiscreteSearchCallback class that handles mapping floating-point variables to discrete values.

__init__(*allowed_values*: List[int | float] = None) → None

Initialization method.

Parameters

allowed_values – Possible values between lower and upper bounds that variables can be mapped.

property allowed_values: List[int | float]

Allowed values between lower and upper bounds.

on_task_begin(*opt_model*: Opyimizer) → None

Performs a callback whenever a task begins.

Parameters

opt_model – An instance of the optimization model.

on_evaluate_before(**evaluate_args*) → None

Performs a callback prior to the *evaluate* method.

7.2 opyimizer.utils.constant

Constants.

7.3 opyoptimizer.utils.exception

Exceptions.

exception opyoptimizer.utils.exception.**Error**(cls: str, msg: str)

A generic Error class derived from Exception.

Essentially, it gets a class object and a message, and logs the error to the logger.

__init__(cls: str, msg: str) → None

Initialization method.

Parameters

- **cls** – Class identifier.
- **msg** – Message to be logged.

exception opyoptimizer.utils.exception.**ArgumentError**(error: str)

An ArgumentError class for logging errors related to wrong number of provided arguments.

__init__(error: str) → None

Initialization method.

Parameters

- **error** – Error message to be logged.

exception opyoptimizer.utils.exception.**BuildError**(error: str)

A BuildError class for logging errors related to classes not being built.

__init__(error: str) → None

Initialization method.

Parameters

- **error** – Error message to be logged.

exception opyoptimizer.utils.exception.**SizeError**(error: str)

A SizeError class for logging errors related to wrong length or size of variables.

__init__(error: str) → None

Initialization method.

Parameters

- **error** – Error message to be logged.

exception opyoptimizer.utils.exception.**TypeError**(error: str)

A TypeError class for logging errors related to wrong type of variables.

__init__(error: str) → None

Initialization method.

Parameters

- **error** – Error message to be logged.

exception opyoptimizer.utils.exception.**ValueError**(error: str)

A ValueError class for logging errors related to wrong value of variables.

__init__(error: str) → None

Initialization method.

Parameters

- **error** – Error message to be logged.

7.4 opyoptimizer.utils.history

History-based object that helps in saving the optimization history.

class opyoptimizer.utils.history.**History**(*save_agents: bool | None = False*)

A History class is responsible for saving each iteration's output.

Note that you can use `dump()` and `parse()` for whatever your needs. Our default is only for agents, best agent and best agent's index.

__init__(*save_agents: bool | None = False*) → None

Initialization method.

Parameters

save_agents – Saves all agents in the search space.

property save_agents: bool

Saves all agents in the search space.

_parse(*key: str, value: Any*) → List[Any] | Tuple[List[Any], float]

Parses incoming values with specified formats.

Parameters

- **key** – Key.
- **value** – Value.

Returns

Parsed value according to the specified format.

Return type

(Union[List[Any], Tuple[List[Any], float]])

dump(***kwargs*) → None

Dumps keyword pairs into self-class attributes.

get_convergence(*key: str, index: Tuple[int, ...] | None = 0*) → ndarray

Gets the convergence list of a specified key.

Parameters

- **key** – Key to be retrieved.
- **index** – Index to be retrieved.

Returns

Values based on key and index.

Return type

(np.ndarray)

7.5 opyoptimizer.utils.logging

Logging-based methods and helpers.

class opyoptimizer.utils.logging.**Logger**(*name, level=0*)

A customized Logger file that enables the possibility of only logging to file.

to_file(*msg: str, *args, **kwargs*) → None

Logs the message only to the logging file.

Parameters

msg – Message to be logged.

opytimizer.utils.logging.get_console_handler() → StreamHandler

Gets a console handler to handle logging into console.

Returns

Handler to output information into console.

Return type

(StreamHandler)

opytimizer.utils.logging.get_timed_file_handler() → TimedRotatingFileHandler

Gets a timed file handler to handle logging into files.

Returns

Handler to output information into timed files.

Return type

(TimedRotatingFileHandler)

opytimizer.utils.logging.get_logger(*logger_name: str*) → *Logger*

Gets a logger and make it available for further use.

Parameters

logger_name – The name of the logger.

Returns

Logger instance.

Return type

(*Logger*)

Utility package for all common opytimizer modules.

OPYTIMIZER.VISUALIZATION

Everyone needs images and plots to help visualize what is happening, correct? This package will provide every visual-related method for you. Check a specific variable convergence, your fitness function convergence, plot benchmark function surfaces, and much more!

8.1 opytimizer.visualization.convergence

Convergence plots.

`opytimizer.visualization.convergence.plot(*args, labels: List[str] | None = None, title: str | None = "", subtitle: str | None = "", xlabel: str | None = 'iteration', ylabel: str | None = 'value', grid: bool | None = True, legend: bool | None = True) → None`

Plots the convergence graph of desired variables.

Essentially, each variable is a list or numpy array with size equals to `n_iterations`.

Parameters

- **labels** – Labels to be applied for each plot in legend.
- **title** – Title of the plot.
- **subtitle** – Subtitle of the plot.
- **xlabel** – Axis *x* label.
- **ylabel** – Axis *y* label.
- **grid** – If grid should be used or not.
- **legend** – If legend should be displayed or not.

8.2 opytimizer.visualization.surface

3-D benchmarking functions plots.

`opytimizer.visualization.surface.plot(points: ndarray, title: str | None = "", subtitle: str | None = "", style: str | None = 'winter', colorbar: bool | None = True) → None`

Plots the surface from a 3-dimensional function.

Parameters

- **points** – Points to be plotted with shape equal to (3, n, n).
- **title** – Title of the plot.
- **subtitle** – Subtitle of the plot.

- **style** – Surface’s style.
- **colorbar** – If colorbar should be used or not.

Visualization package for all common opyimizer modules.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

O

- `opytimizer.core`, 13
- `opytimizer.core.agent`, 5
- `opytimizer.core.block`, 6
- `opytimizer.core.cell`, 7
- `opytimizer.core.function`, 8
- `opytimizer.core.node`, 9
- `opytimizer.core.optimizer`, 11
- `opytimizer.core.space`, 12
- `opytimizer.functions`, 17
- `opytimizer.functions.constrained`, 15
- `opytimizer.functions.multi_objective`, 17
- `opytimizer.functions.multi_objective.standard`, 16
- `opytimizer.functions.multi_objective.weighted`, 16
- `opytimizer.math`, 23
- `opytimizer.math.distribution`, 19
- `opytimizer.math.general`, 20
- `opytimizer.math.random`, 21
- `opytimizer.optimizers`, 124
- `opytimizer.optimizers.boolean`, 28
- `opytimizer.optimizers.boolean.bmrfo`, 25
- `opytimizer.optimizers.boolean.bpso`, 26
- `opytimizer.optimizers.boolean.umda`, 27
- `opytimizer.optimizers.evolutionary`, 44
- `opytimizer.optimizers.evolutionary.bsa`, 28
- `opytimizer.optimizers.evolutionary.de`, 29
- `opytimizer.optimizers.evolutionary.ep`, 30
- `opytimizer.optimizers.evolutionary.es`, 32
- `opytimizer.optimizers.evolutionary.foa`, 33
- `opytimizer.optimizers.evolutionary.ga`, 34
- `opytimizer.optimizers.evolutionary.gp`, 35
- `opytimizer.optimizers.evolutionary.hs`, 37
- `opytimizer.optimizers.evolutionary.iwo`, 42
- `opytimizer.optimizers.evolutionary.rra`, 43
- `opytimizer.optimizers.misc`, 49
- `opytimizer.optimizers.misc.aoa`, 44
- `opytimizer.optimizers.misc.cem`, 45
- `opytimizer.optimizers.misc.doa`, 46
- `opytimizer.optimizers.misc.gs`, 47
- `opytimizer.optimizers.misc.hc`, 47
- `opytimizer.optimizers.misc.nds`, 48
- `opytimizer.optimizers.population`, 65
- `opytimizer.optimizers.population.aeo`, 49
- `opytimizer.optimizers.population.ao`, 51
- `opytimizer.optimizers.population.coa`, 52
- `opytimizer.optimizers.population.epo`, 53
- `opytimizer.optimizers.population.gco`, 53
- `opytimizer.optimizers.population.gwo`, 55
- `opytimizer.optimizers.population.hho`, 55
- `opytimizer.optimizers.population.loa`, 57
- `opytimizer.optimizers.population.osa`, 61
- `opytimizer.optimizers.population.ppa`, 62
- `opytimizer.optimizers.population.pvs`, 63
- `opytimizer.optimizers.population.rfo`, 64
- `opytimizer.optimizers.science`, 83
- `opytimizer.optimizers.science.aig`, 65
- `opytimizer.optimizers.science.aso`, 65
- `opytimizer.optimizers.science.bh`, 67
- `opytimizer.optimizers.science.efo`, 68
- `opytimizer.optimizers.science.eo`, 69
- `opytimizer.optimizers.science.esa`, 70
- `opytimizer.optimizers.science.gsa`, 71
- `opytimizer.optimizers.science.hgso`, 72
- `opytimizer.optimizers.science.lsa`, 73
- `opytimizer.optimizers.science.moa`, 74
- `opytimizer.optimizers.science.mvo`, 75
- `opytimizer.optimizers.science.sa`, 76
- `opytimizer.optimizers.science.teo`, 76
- `opytimizer.optimizers.science.two`, 77
- `opytimizer.optimizers.science.wca`, 78
- `opytimizer.optimizers.science.wdo`, 80
- `opytimizer.optimizers.science.weo`, 80
- `opytimizer.optimizers.science.wwo`, 81
- `opytimizer.optimizers.social`, 90
- `opytimizer.optimizers.social.bso`, 83
- `opytimizer.optimizers.social.ci`, 84
- `opytimizer.optimizers.social.isa`, 85
- `opytimizer.optimizers.social.mvpa`, 86
- `opytimizer.optimizers.social.qsa`, 87
- `opytimizer.optimizers.social.ssd`, 88
- `opytimizer.optimizers.swarm`, 124
- `opytimizer.optimizers.swarm.abc`, 90

- opytimizer.optimizers.swarm.abo, 91
- opytimizer.optimizers.swarm.af, 92
- opytimizer.optimizers.swarm.ba, 93
- opytimizer.optimizers.swarm.boa, 94
- opytimizer.optimizers.swarm.bwo, 95
- opytimizer.optimizers.swarm.cs, 96
- opytimizer.optimizers.swarm.csa, 98
- opytimizer.optimizers.swarm.eho, 99
- opytimizer.optimizers.swarm.fa, 100
- opytimizer.optimizers.swarm.ffoa, 100
- opytimizer.optimizers.swarm.fpa, 101
- opytimizer.optimizers.swarm.fso, 102
- opytimizer.optimizers.swarm.goa, 103
- opytimizer.optimizers.swarm.js, 104
- opytimizer.optimizers.swarm.kh, 106
- opytimizer.optimizers.swarm.mfo, 111
- opytimizer.optimizers.swarm.mrfo, 111
- opytimizer.optimizers.swarm.pio, 113
- opytimizer.optimizers.swarm.pso, 114
- opytimizer.optimizers.swarm.sbo, 117
- opytimizer.optimizers.swarm.sca, 118
- opytimizer.optimizers.swarm.sfo, 119
- opytimizer.optimizers.swarm.sos, 120
- opytimizer.optimizers.swarm.ssa, 121
- opytimizer.optimizers.swarm.sso, 122
- opytimizer.optimizers.swarm.stoa, 123
- opytimizer.optimizers.swarm.woa, 123
- opytimizer.spaces, 129
 - opytimizer.spaces.boolean, 125
 - opytimizer.spaces.graph, 125
 - opytimizer.spaces.grid, 125
 - opytimizer.spaces.hyper_complex, 126
 - opytimizer.spaces.pareto, 126
 - opytimizer.spaces.search, 127
 - opytimizer.spaces.tree, 127
- opytimizer.utils, 136
 - opytimizer.utils.callback, 131
 - opytimizer.utils.constant, 133
 - opytimizer.utils.exception, 134
 - opytimizer.utils.history, 135
 - opytimizer.utils.logging, 135
- opytimizer.visualization, 138
 - opytimizer.visualization.convergence, 137
 - opytimizer.visualization.surface, 137

Symbols

<code>__call__()</code> (<i>optimizer.core.block.Block</i> method), 6	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.ep.EP</i> method), 31
<code>__call__()</code> (<i>optimizer.core.cell.Cell</i> method), 7	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.es.ES</i> method), 32
<code>__call__()</code> (<i>optimizer.core.function.Function</i> method), 8	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.foa.FOA</i> method), 33
<code>__call__()</code> (<i>optimizer.functions.constrained.ConstrainedFunction</i> method), 15	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.ga.GA</i> method), 34
<code>__call__()</code> (<i>optimizer.functions.multi_objective.standard.MultiObjectiveFunction</i> method), 16	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.gp.GP</i> method), 36
<code>__call__()</code> (<i>optimizer.functions.multi_objective.weighted.MultiObjectiveWeightedFunction</i> method), 16	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.GHS</i> method), 39
<code>__init__()</code> (<i>optimizer.Optimizer</i> method), 3	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.GOGHS</i> method), 41
<code>__init__()</code> (<i>optimizer.core.agent.Agent</i> method), 5	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.HS</i> method), 37
<code>__init__()</code> (<i>optimizer.core.block.Block</i> method), 6	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.IHS</i> method), 38
<code>__init__()</code> (<i>optimizer.core.block.InnerBlock</i> method), 7	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.NGHS</i> method), 40
<code>__init__()</code> (<i>optimizer.core.block.InputBlock</i> method), 7	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.hs.SGHS</i> method), 39
<code>__init__()</code> (<i>optimizer.core.block.OutputBlock</i> method), 7	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.iwo.IWO</i> method), 42
<code>__init__()</code> (<i>optimizer.core.cell.Cell</i> method), 7	<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.rra.RRA</i> method), 43
<code>__init__()</code> (<i>optimizer.core.function.Function</i> method), 8	<code>__init__()</code> (<i>optimizer.optimizers.misc.aoa.AOA</i> method), 44
<code>__init__()</code> (<i>optimizer.core.node.Node</i> method), 9	<code>__init__()</code> (<i>optimizer.optimizers.misc.cem.CEM</i> method), 45
<code>__init__()</code> (<i>optimizer.core.optimizer.Optimizer</i> method), 11	<code>__init__()</code> (<i>optimizer.optimizers.misc.doa.DOA</i> method), 46
<code>__init__()</code> (<i>optimizer.core.space.Space</i> method), 12	<code>__init__()</code> (<i>optimizer.optimizers.misc.gs.GS</i> method), 47
<code>__init__()</code> (<i>optimizer.functions.constrained.ConstrainedFunction</i> method), 15	<code>__init__()</code> (<i>optimizer.optimizers.misc.hc.HC</i> method), 48
<code>__init__()</code> (<i>optimizer.functions.multi_objective.standard.MultiObjectiveFunction</i> method), 16	<code>__init__()</code> (<i>optimizer.optimizers.misc.nds.NDS</i> method), 48
<code>__init__()</code> (<i>optimizer.functions.multi_objective.weighted.MultiObjectiveWeightedFunction</i> method), 16	<code>__init__()</code> (<i>optimizer.optimizers.population.aeo.AEO</i> method), 49
<code>__init__()</code> (<i>optimizer.optimizers.boolean.bmrfo.BMRFO</i> method), 25	<code>__init__()</code> (<i>optimizer.optimizers.population.ao.AO</i> method), 49
<code>__init__()</code> (<i>optimizer.optimizers.boolean.bps.BPSO</i> method), 26	
<code>__init__()</code> (<i>optimizer.optimizers.boolean.uda.UMDA</i> method), 27	
<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.bsa.BSA</i> method), 28	
<code>__init__()</code> (<i>optimizer.optimizers.evolutionary.de.DE</i> method), 30	

method), 51

`__init__()` (optimizer.optimizers.population.coa.COA method), 52

`__init__()` (optimizer.optimizers.population.epo.EPO method), 53

`__init__()` (optimizer.optimizers.population.gco.GCO method), 54

`__init__()` (optimizer.optimizers.population.gwo.GWO method), 55

`__init__()` (optimizer.optimizers.population.hho.HHO method), 56

`__init__()` (optimizer.optimizers.population.loa.LOA method), 58

`__init__()` (optimizer.optimizers.population.loa.Lion method), 57

`__init__()` (optimizer.optimizers.population.osa.OSA method), 61

`__init__()` (optimizer.optimizers.population.ppa.PPA method), 62

`__init__()` (optimizer.optimizers.population.pvs.PVS method), 63

`__init__()` (optimizer.optimizers.population.rfo.RFO method), 64

`__init__()` (optimizer.optimizers.science.aig.AIG method), 65

`__init__()` (optimizer.optimizers.science.aso.ASO method), 66

`__init__()` (optimizer.optimizers.science.bh.BH method), 67

`__init__()` (optimizer.optimizers.science.efo.EFO method), 68

`__init__()` (optimizer.optimizers.science.eo.EO method), 69

`__init__()` (optimizer.optimizers.science.esa.ESA method), 70

`__init__()` (optimizer.optimizers.science.gsa.GSA method), 71

`__init__()` (optimizer.optimizers.science.hgso.HGSO method), 72

`__init__()` (optimizer.optimizers.science.lsa.LSA method), 73

`__init__()` (optimizer.optimizers.science.moa.MOA method), 75

`__init__()` (optimizer.optimizers.science.mvo.MVO method), 75

`__init__()` (optimizer.optimizers.science.sa.SA method), 76

`__init__()` (optimizer.optimizers.science.teo.TEO method), 77

`__init__()` (optimizer.optimizers.science.two.TWO method), 77

`__init__()` (optimizer.optimizers.science.wca.WCA method), 78

`__init__()` (optimizer.optimizers.science.wdo.WDO method), 80

`__init__()` (optimizer.optimizers.science.weo.WEO method), 81

`__init__()` (optimizer.optimizers.science.wwc.WWC method), 82

`__init__()` (optimizer.optimizers.social.bso.BSO method), 83

`__init__()` (optimizer.optimizers.social.ci.CI method), 85

`__init__()` (optimizer.optimizers.social.isa.ISA method), 85

`__init__()` (optimizer.optimizers.social.mvpa.MVPA method), 86

`__init__()` (optimizer.optimizers.social.qsa.QSA method), 87

`__init__()` (optimizer.optimizers.social.ssd.SSD method), 88

`__init__()` (optimizer.optimizers.swarm.abc.ABC method), 90

`__init__()` (optimizer.optimizers.swarm.abo.ABO method), 91

`__init__()` (optimizer.optimizers.swarm.af.AF method), 92

`__init__()` (optimizer.optimizers.swarm.ba.BA method), 93

`__init__()` (optimizer.optimizers.swarm.boa.BOA method), 94

`__init__()` (optimizer.optimizers.swarm.bwo.BWO method), 96

`__init__()` (optimizer.optimizers.swarm.cs.CS method), 97

`__init__()` (optimizer.optimizers.swarm.csa.CSA method), 98

`__init__()` (optimizer.optimizers.swarm.eho.EHO method), 99

`__init__()` (optimizer.optimizers.swarm.fa.FA method), 100

`__init__()` (optimizer.optimizers.swarm.ffoa.FFOA method), 101

`__init__()` (optimizer.optimizers.swarm.fpa.FPA method), 101

`__init__()` (optimizer.optimizers.swarm.fso.FSO method), 102

`__init__()` (optimizer.optimizers.swarm.goa.GOA method), 103

`__init__()` (optimizer.optimizers.swarm.js.JS method), 104

`__init__()` (optimizer.optimizers.swarm.js.NBJS method), 105

`__init__()` (optimizer.optimizers.swarm.kh.KH method), 106

`__init__()` (optimizer.optimizers.swarm.mfo.MFO method), 111

`__init__()` (optimizer.optimizers.swarm.mrfo.MRFO method), 111

`method`), 111
`__init__()` (`opyoptimizer.optimizers.swarm.pio.PIO` `method`), 113
`__init__()` (`opyoptimizer.optimizers.swarm.pso.AIWPSO` `method`), 115
`__init__()` (`opyoptimizer.optimizers.swarm.pso.PSO` `method`), 114
`__init__()` (`opyoptimizer.optimizers.swarm.pso.RPSO` `method`), 116
`__init__()` (`opyoptimizer.optimizers.swarm.pso.SAVPSO` `method`), 116
`__init__()` (`opyoptimizer.optimizers.swarm.pso.VPSO` `method`), 116
`__init__()` (`opyoptimizer.optimizers.swarm.sbo.SBO` `method`), 117
`__init__()` (`opyoptimizer.optimizers.swarm.sca.SCA` `method`), 118
`__init__()` (`opyoptimizer.optimizers.swarm.sfo.SFO` `method`), 119
`__init__()` (`opyoptimizer.optimizers.swarm.sos.SOS` `method`), 120
`__init__()` (`opyoptimizer.optimizers.swarm.ssa.SSA` `method`), 121
`__init__()` (`opyoptimizer.optimizers.swarm.sso.SSO` `method`), 122
`__init__()` (`opyoptimizer.optimizers.swarm.stoa.STOA` `method`), 123
`__init__()` (`opyoptimizer.optimizers.swarm.woa.WOA` `method`), 123
`__init__()` (`opyoptimizer.spaces.boolean.BooleanSpace` `method`), 125
`__init__()` (`opyoptimizer.spaces.grid.GridSpace` `method`), 125
`__init__()` (`opyoptimizer.spaces.hyper_complex.HyperComplexSpace` `method`), 126
`__init__()` (`opyoptimizer.spaces.pareto.ParetoSpace` `method`), 126
`__init__()` (`opyoptimizer.spaces.search.SearchSpace` `method`), 127
`__init__()` (`opyoptimizer.spaces.tree.TreeSpace` `method`), 128
`__init__()` (`opyoptimizer.utils.callback.Callback` `method`), 131
`__init__()` (`opyoptimizer.utils.callback.CallbackVessel` `method`), 132
`__init__()` (`opyoptimizer.utils.callback.CheckpointCallback` `method`), 133
`__init__()` (`opyoptimizer.utils.callback.DiscreteSearchCallback` `method`), 133
`__init__()` (`opyoptimizer.utils.exception.ArgumentError` `method`), 134
`__init__()` (`opyoptimizer.utils.exception.BuildError` `method`), 134
`__init__()` (`opyoptimizer.utils.exception.Error` `method`), 134
`__init__()` (`opyoptimizer.utils.exception.SizeError` `method`), 134
`__init__()` (`opyoptimizer.utils.exception.TypeError` `method`), 134
`__init__()` (`opyoptimizer.utils.exception.ValueError` `method`), 134
`__init__()` (`opyoptimizer.utils.history.History` `method`), 135
`__repr__()` (`opyoptimizer.core.node.Node` `method`), 9
`__str__()` (`opyoptimizer.core.node.Node` `method`), 9
`_average_concentration()` (`opyoptimizer.optimizers.science.eo.EO` `method`), 69
`_best_beta()` (`opyoptimizer.optimizers.swarm.kh.KH` `method`), 109
`_best_movement()` (`opyoptimizer.optimizers.swarm.boa.BOA` `method`), 95
`_break_wave()` (`opyoptimizer.optimizers.science.wwo.WWO` `method`), 83
`_build_string()` (in module `opyoptimizer.core.node`), 10
`_business_one()` (`opyoptimizer.optimizers.social.qsa.QSA` `method`), 88
`_business_three()` (`opyoptimizer.optimizers.social.qsa.QSA` `method`), 88
`_business_two()` (`opyoptimizer.optimizers.social.qsa.QSA` `method`), 88
`_calculate_acceleration()` (`opyoptimizer.optimizers.science.aso.ASO` `method`),
`_calculate_center()` (`opyoptimizer.optimizers.swarm.pio.PIO` `method`), 113
`_calculate_chaotic_map()` (`opyoptimizer.optimizers.misc.doa.DOA` `method`), 47
`_calculate_coefficients()` (`opyoptimizer.optimizers.population.gwo.GWO` `method`), 55
`_calculate_equilibrium()` (`opyoptimizer.optimizers.science.eo.EO` `method`), 69
`_calculate_force()` (`opyoptimizer.optimizers.science.gsa.GSA` `method`), 71
`_calculate_indexes()` (`opyoptimizer.optimizers.science.efo.EFO` `method`), 68
`_calculate_initial_coefficients()` (`opyoptimizer.optimizers.population.hho.HHO`

- method*), 56
- `_calculate_lambda_i()` (*optimizer.optimizers.swarm.sfo.SFO method*), 119
- `_calculate_mass()` (*optimizer.optimizers.science.aso.ASO method*), 66
- `_calculate_mass()` (*optimizer.optimizers.science.gsa.GSA method*), 71
- `_calculate_population()` (*optimizer.optimizers.population.ppa.PPA method*), 62
- `_calculate_potential()` (*optimizer.optimizers.science.aso.ASO method*), 66
- `_calculate_probability()` (*optimizer.optimizers.boolean.umda.UMDA method*), 28
- `_calculate_queue()` (*optimizer.optimizers.social.qsa.QSA method*), 87
- `_carnivore_consumption()` (*optimizer.optimizers.population.aeo.AEO method*), 50
- `_chain_foraging()` (*optimizer.optimizers.boolean.bmrfo.BMRFO method*), 26
- `_chain_foraging()` (*optimizer.optimizers.swarm.mrfo.MRFO method*), 112
- `_check_prides_for_males()` (*optimizer.optimizers.population.loa.LOA method*), 61
- `_clusterize()` (*optimizer.optimizers.social.bso.BSO method*), 84
- `_commensalism()` (*optimizer.optimizers.swarm.sos.SOS method*), 121
- `_compare_domination()` (*optimizer.optimizers.misc.nds.NDS method*), 48
- `_compute_success()` (*optimizer.optimizers.swarm.pso.AIWPSO method*), 115
- `_constraint_handle()` (*optimizer.optimizers.science.two.TWO method*), 78
- `_create_agents()` (*optimizer.core.space.Space method*), 12
- `_create_grid()` (*optimizer.spaces.grid.GridSpace method*), 126
- `_create_new_samples()` (*optimizer.optimizers.misc.cem.CEM method*), 45
- `_create_terminals()` (*optimizer.spaces.tree.TreeSpace method*), 128
- `_create_trees()` (*optimizer.spaces.tree.TreeSpace method*), 128
- `_cross()` (*optimizer.optimizers.evolutionary.gp.GP method*), 37
- `_crossover()` (*optimizer.optimizers.evolutionary.bsa.BSA method*), 29
- `_crossover()` (*optimizer.optimizers.evolutionary.ga.GA method*), 35
- `_crossover()` (*optimizer.optimizers.evolutionary.gp.GP method*), 36
- `_crossover()` (*optimizer.optimizers.swarm.kh.KH method*), 110
- `_cyclone_foraging()` (*optimizer.optimizers.boolean.bmrfo.BMRFO method*), 25
- `_cyclone_foraging()` (*optimizer.optimizers.swarm.mrfo.MRFO method*), 112
- `_dark_zone()` (*optimizer.optimizers.population.gco.GCO method*), 54
- `_defense()` (*optimizer.optimizers.population.loa.LOA method*), 59
- `_equilibrium()` (*optimizer.optimizers.population.loa.LOA method*), 61
- `_evaluate()` (*in module optimizer.core.node*), 10
- `_evaluate_location()` (*optimizer.optimizers.swarm.abc.ABC method*), 91
- `_evaluate_nests()` (*optimizer.optimizers.swarm.cs.CS method*), 97
- `_evaporation_flux()` (*optimizer.optimizers.science.weo.WEO method*), 81
- `_event_horizon()` (*optimizer.optimizers.science.bh.BH method*), 67
- `_exploitation_phase()` (*optimizer.optimizers.population.hho.HHO method*), 56
- `_exploration_phase()` (*optimizer.optimizers.population.hho.HHO method*), 56
- `_flight_mode()` (*optimizer.optimizers.swarm.abo.ABO method*), 92
- `_flow_intensity()` (*optimizer.optimizers.science.wca.WCA method*), 79
- `_food_beta()` (*optimizer.optimizers.swarm.kh.KH*

<i>method</i>), 108			
<code>_food_location()</code>	(<i>opy-timizer.optimizers.swarm.kh.KH</i>	<i>method</i>),	
107			
<code>_foraging_motion()</code>	(<i>opy-timizer.optimizers.swarm.kh.KH</i>	<i>method</i>),	
109			
<code>_generate_abandoned_nests()</code>	(<i>opy-timizer.optimizers.swarm.cs.CS</i>	<i>method</i>),	
97			
<code>_generate_new_harmony()</code>	(<i>opy-timizer.optimizers.evolutionary.hs.GHS</i>	<i>method</i>),	39
<code>_generate_new_harmony()</code>	(<i>opy-timizer.optimizers.evolutionary.hs.HS</i>	<i>method</i>),	38
<code>_generate_new_harmony()</code>	(<i>opy-timizer.optimizers.evolutionary.hs.NGHS</i>	<i>method</i>),	41
<code>_generate_new_harmony()</code>	(<i>opy-timizer.optimizers.evolutionary.hs.SGHS</i>	<i>method</i>),	40
<code>_generate_new_nests()</code>	(<i>opy-timizer.optimizers.swarm.cs.CS</i>	<i>method</i>),	97
<code>_generate_opposition_harmony()</code>	(<i>opy-timizer.optimizers.evolutionary.hs.GOGHS</i>	<i>method</i>),	41
<code>_generate_random_agent()</code>	(<i>opy-timizer.optimizers.swarm.sfo.SFO</i>	<i>method</i>),	119
<code>_generate_random_agent()</code>	(<i>opy-timizer.optimizers.swarm.woa.WOA</i>	<i>method</i>),	123
<code>_get_agents_from_clan()</code>	(<i>opy-timizer.optimizers.swarm.eho.EHO</i>	<i>method</i>),	99
<code>_get_agents_from_pack()</code>	(<i>opy-timizer.optimizers.population.coa.COA</i>	<i>method</i>),	52
<code>_get_agents_from_team()</code>	(<i>opy-timizer.optimizers.social.mvpa.MVPA</i>	<i>method</i>),	87
<code>_get_neighbours()</code>	(<i>opy-timizer.optimizers.swarm.kh.KH</i>	<i>method</i>),	107
<code>_get_nomad_lions()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	58
<code>_get_pride_lions()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	58
<code>_global_pollination()</code>	(<i>opy-timizer.optimizers.swarm.fpa.FPA</i>	<i>method</i>),	
			101
<code>_global_seeding()</code>	(<i>opy-timizer.optimizers.evolutionary.foa.FOA</i>	<i>method</i>),	34
<code>_herbivore_consumption()</code>	(<i>opy-timizer.optimizers.population.aeo.AEO</i>	<i>method</i>),	50
<code>_hunting()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	59
<code>_initialize_agents()</code>	(<i>opy-timizer.core.space.Space</i>	<i>method</i>),	13
<code>_initialize_agents()</code>	(<i>opy-timizer.spaces.boolean.BooleanSpace</i>	<i>method</i>),	125
<code>_initialize_agents()</code>	(<i>opy-timizer.spaces.grid.GridSpace</i>	<i>method</i>),	126
<code>_initialize_agents()</code>	(<i>opy-timizer.spaces.hyper_complex.HyperComplexSpace</i>	<i>method</i>),	126
<code>_initialize_agents()</code>	(<i>opy-timizer.spaces.search.SearchSpace</i>	<i>method</i>),	127
<code>_initialize_agents()</code>	(<i>opy-timizer.spaces.tree.TreeSpace</i>	<i>method</i>),	128
<code>_initialize_chaotic_map()</code>	(<i>opy-timizer.optimizers.swarm.js.JS</i>	<i>method</i>),	104
<code>_initialize_terminals()</code>	(<i>opy-timizer.spaces.tree.TreeSpace</i>	<i>method</i>),	128
<code>_light_zone()</code>	(<i>opy-timizer.optimizers.population.gco.GCO</i>	<i>method</i>),	54
<code>_load_agents()</code>	(<i>opy-timizer.spaces.pareto.ParetoSpace</i>	<i>method</i>),	127
<code>_local_alpha()</code>	(<i>opy-timizer.optimizers.swarm.kh.KH</i>	<i>method</i>),	107
<code>_local_movement()</code>	(<i>opy-timizer.optimizers.swarm.boa.BOA</i>	<i>method</i>),	95
<code>_local_pollination()</code>	(<i>opy-timizer.optimizers.swarm.fpa.FPA</i>	<i>method</i>),	102
<code>_local_seeding()</code>	(<i>opy-timizer.optimizers.evolutionary.foa.FOA</i>	<i>method</i>),	33
<code>_mating()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	59
<code>_mating_operator()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	59
<code>_mean_global_solution()</code>	(<i>opy-timizer.optimizers.social.ssd.SSD</i>	<i>method</i>),	89
<code>_migrating()</code>	(<i>opy-timizer.optimizers.population.loa.LOA</i>	<i>method</i>),	60

<code>_motion_a()</code>	(<code>optimizer.optimizers.swarm.js.JS</code> method), 105	<code>_omnivore_consumption()</code>	(<code>optimizer.optimizers.population.aeo.AEO</code> method), 50
<code>_motion_a()</code>	(<code>optimizer.optimizers.swarm.js.NBJS</code> method), 105	<code>_parasitism()</code>	(<code>optimizer.optimizers.swarm.sos.SOS</code> method), 121
<code>_motion_b()</code>	(<code>optimizer.optimizers.swarm.js.JS</code> method), 105	<code>_parasitism_phase()</code>	(<code>optimizer.optimizers.population.ppa.PPA</code> method), 63
<code>_moving_safe_place()</code>	(<code>optimizer.optimizers.population.loa.LOA</code> method), 59	<code>_parse()</code>	(<code>optimizer.utils.history.History</code> method), 135
<code>_mutate()</code>	(<code>optimizer.optimizers.evolutionary.bsa.BSA</code> method), 29	<code>_permute()</code>	(<code>optimizer.optimizers.evolutionary.bsa.BSA</code> method), 29
<code>_mutate()</code>	(<code>optimizer.optimizers.evolutionary.gp.GP</code> method), 36	<code>_physical_diffusion()</code>	(<code>optimizer.optimizers.swarm.kh.KH</code> method), 109
<code>_mutate_agent()</code>	(<code>optimizer.optimizers.evolutionary.de.DE</code> method), 30	<code>_population_limiting()</code>	(<code>optimizer.optimizers.evolutionary.foa.FOA</code> method), 34
<code>_mutate_cell()</code>	(<code>optimizer.optimizers.population.gco.GCO</code> method), 54	<code>_predation_phase()</code>	(<code>optimizer.optimizers.population.ppa.PPA</code> method), 63
<code>_mutate_parent()</code>	(<code>optimizer.optimizers.evolutionary.ep.EP</code> method), 31	<code>_procreating()</code>	(<code>optimizer.optimizers.swarm.bwo.BWO</code> method), 96
<code>_mutate_parent()</code>	(<code>optimizer.optimizers.evolutionary.es.ES</code> method), 32	<code>_produce_offspring()</code>	(<code>optimizer.optimizers.evolutionary.iwo.IWO</code> method), 42
<code>_mutation()</code>	(<code>optimizer.optimizers.evolutionary.ga.GA</code> method), 35	<code>_production()</code>	(<code>optimizer.optimizers.population.aeo.AEO</code> method), 49
<code>_mutation()</code>	(<code>optimizer.optimizers.evolutionary.gp.GP</code> method), 36	<code>_propagate_wave()</code>	(<code>optimizer.optimizers.science.wwo.WWO</code> method), 82
<code>_mutation()</code>	(<code>optimizer.optimizers.swarm.bwo.BWO</code> method), 96	<code>_properties()</code>	(in module <code>optimizer.core.node</code>), 10
<code>_mutation()</code>	(<code>optimizer.optimizers.swarm.kh.KH</code> method), 110	<code>_prune_nodes()</code>	(<code>optimizer.optimizers.evolutionary.gp.GP</code> method), 36
<code>_mutualism()</code>	(<code>optimizer.optimizers.swarm.sos.SOS</code> method), 120	<code>_raining_process()</code>	(<code>optimizer.optimizers.science.wca.WCA</code> method), 79
<code>_neighbour_motion()</code>	(<code>optimizer.optimizers.swarm.kh.KH</code> method), 108	<code>_refract_wave()</code>	(<code>optimizer.optimizers.science.wwo.WWO</code> method), 82
<code>_nesting_phase()</code>	(<code>optimizer.optimizers.population.ppa.PPA</code> method), 62	<code>_relocation()</code>	(<code>optimizer.optimizers.population.rfo.RFO</code> method), 64
<code>_nomad_attack()</code>	(<code>optimizer.optimizers.population.loa.LOA</code> method), 60	<code>_reproduction()</code>	(<code>optimizer.optimizers.evolutionary.gp.GP</code> method), 36
<code>_nomad_mating()</code>	(<code>optimizer.optimizers.population.loa.LOA</code> method), 60	<code>_roaming()</code>	(<code>optimizer.optimizers.population.loa.LOA</code> method), 59
<code>_nomad_roaming()</code>	(<code>optimizer.optimizers.population.loa.LOA</code> method), 60	<code>_roulette_selection()</code>	(<code>optimizer.optimizers.evolutionary.ga.GA</code> method), 35
<code>_noticing()</code>	(<code>optimizer.optimizers.population.rfo.RFO</code> method), 64	<code>_roulette_selection()</code>	(<code>optimizer.optimizers.evolutionary.rra.RRA</code> method), 35
<code>_ocean_current()</code>	(<code>optimizer.optimizers.swarm.js.JS</code> method), 104		

<i>method</i>), 44		<i>_update_position()</i> (<i>opytimizer.optimizers.science.bh.BH</i> <i>method</i>), 67
<i>_sample_position()</i> (<i>opytimizer.optimizers.boolean.umd.UMDA</i> <i>method</i>), 28		<i>_update_position()</i> (<i>opytimizer.optimizers.science.hgso.HGSO</i> <i>method</i>), 73
<i>_send_employee()</i> (<i>opytimizer.optimizers.swarm.abc.ABC</i> <i>method</i>), 91		<i>_update_position()</i> (<i>opytimizer.optimizers.science.lsa.LSA</i> <i>method</i>), 74
<i>_send_onlooker()</i> (<i>opytimizer.optimizers.swarm.abc.ABC</i> <i>method</i>), 91		<i>_update_position()</i> (<i>opytimizer.optimizers.social.ssd.SSD</i> <i>method</i>), 89
<i>_send_scout()</i> (<i>opytimizer.optimizers.swarm.abc.ABC</i> <i>method</i>), 91		<i>_update_position()</i> (<i>opytimizer.optimizers.swarm.kh.KH</i> <i>method</i>), 110
<i>_sensing_distance()</i> (<i>opytimizer.optimizers.swarm.kh.KH</i> <i>method</i>), 107		<i>_update_position()</i> (<i>opytimizer.optimizers.swarm.sca.SCA</i> <i>method</i>), 118
<i>_separating_operator()</i> (<i>opytimizer.optimizers.swarm.eho.EHO</i> <i>method</i>), 99		<i>_update_river()</i> (<i>opytimizer.optimizers.science.wca.WCA</i> <i>method</i>), 79
<i>_sigmoid()</i> (<i>opytimizer.optimizers.social.bso.BSO</i> <i>method</i>), 84		<i>_update_sailfish()</i> (<i>opytimizer.optimizers.swarm.sfo.SFO</i> <i>method</i>), 120
<i>_social_force()</i> (<i>opytimizer.optimizers.swarm.goa.GOA</i> <i>method</i>), 103		<i>_update_std()</i> (<i>opytimizer.optimizers.misc.cem.CEM</i> <i>method</i>), 46
<i>_somersault_foraging()</i> (<i>opytimizer.optimizers.boolean.bmrfo.BMRFO</i> <i>method</i>), 26		<i>_update_strategy()</i> (<i>opytimizer.optimizers.evolutionary.ep.EP</i> <i>method</i>), 31
<i>_somersault_foraging()</i> (<i>opytimizer.optimizers.swarm.mrfo.MRFO</i> <i>method</i>), 112		<i>_update_strategy()</i> (<i>opytimizer.optimizers.evolutionary.es.ES</i> <i>method</i>), 32
<i>_spatial_dispersal()</i> (<i>opytimizer.optimizers.evolutionary.iwo.IWO</i> <i>method</i>), 42		<i>_update_stream()</i> (<i>opytimizer.optimizers.science.wca.WCA</i> <i>method</i>), 79
<i>_stalling_search()</i> (<i>opytimizer.optimizers.evolutionary.rra.RRA</i> <i>method</i>), 43		<i>_update_velocity()</i> (<i>opytimizer.optimizers.social.ssd.SSD</i> <i>method</i>), 89
<i>_target_alpha()</i> (<i>opytimizer.optimizers.swarm.kh.KH</i> <i>method</i>), 108		<i>_update_wave_length()</i> (<i>opytimizer.optimizers.science.ww. WWO</i> <i>method</i>), 83
<i>_transition_packs()</i> (<i>opytimizer.optimizers.population.coa.COA</i> <i>method</i>), 52		<i>_updating_operator()</i> (<i>opytimizer.optimizers.swarm.eho.EHO</i> <i>method</i>), 99
<i>_update_center_position()</i> (<i>opytimizer.optimizers.swarm.pio.PIO</i> <i>method</i>), 113		
<i>_update_composition()</i> (<i>opytimizer.optimizers.population.aeo.AEO</i> <i>method</i>), 50		
<i>_update_decomposition()</i> (<i>opytimizer.optimizers.population.aeo.AEO</i> <i>method</i>), 51		
<i>_update_direction()</i> (<i>opytimizer.optimizers.science.lsa.LSA</i> <i>method</i>), 74		
<i>_update_mean()</i> (<i>opytimizer.optimizers.misc.cem.CEM</i> <i>method</i>), 46		

A

opytimizer.optimizers.swarm.abo.ABO property), 92
A (*opytimizer.optimizers.swarm.ba.BA* property), 93
a (*opytimizer.optimizers.swarm.boa.BOA* property), 94
a (*opytimizer.optimizers.swarm.sca.SCA* property), 118
A (*opytimizer.optimizers.swarm.sfo.SFO* property), 119
a1 (*opytimizer.optimizers.science.eo.EO* property), 69
a2 (*opytimizer.optimizers.science.eo.EO* property), 69
a_max (*opytimizer.optimizers.misc.aoa.AOA* property), 44

- `a_min` (`opytimize.optimizer.misc.aoa.AOA` property), 44
- `ABC` (class in `opytimize.optimizer.swarm.abc`), 90
- `ABO` (class in `opytimize.optimizer.swarm.abo`), 91
- `AE0` (class in `opytimize.optimizer.population.aeo`), 49
- `AF` (class in `opytimize.optimizer.swarm.af`), 92
- `age` (`opytimize.optimizer.evolutionary.foa.FOA` property), 33
- `Agent` (class in `opytimize.core.agent`), 5
- `agents` (`opytimize.core.space.Space` property), 12
- `AIG` (class in `opytimize.optimizer.science.aig`), 65
- `AIWPSO` (class in `opytimize.optimizer.swarm.pso`), 115
- `algorithm` (`opytimize.core.optimizer.Optimizer` property), 11
- `allowed_values` (`opytimize.utils.callback.DiscreteSearchCallback` property), 133
- `alpha` (`opytimize.optimizer.misc.aoa.AOA` property), 45
- `alpha` (`opytimize.optimizer.misc.cem.CEM` property), 45
- `alpha` (`opytimize.optimizer.population.ao.AO` property), 51
- `alpha` (`opytimize.optimizer.science.aig.AIG` property), 65
- `alpha` (`opytimize.optimizer.science.aso.ASO` property), 66
- `alpha` (`opytimize.optimizer.science.hgso.HGSO` property), 72
- `alpha` (`opytimize.optimizer.science.moa.MOA` property), 75
- `alpha` (`opytimize.optimizer.science.two.TWO` property), 78
- `alpha` (`opytimize.optimizer.science.wdo.WDO` property), 80
- `alpha` (`opytimize.optimizer.science.ww0.WWO` property), 82
- `alpha` (`opytimize.optimizer.swarm.cs.CS` property), 97
- `alpha` (`opytimize.optimizer.swarm.eho.EHO` property), 99
- `alpha` (`opytimize.optimizer.swarm.fa.FA` property), 100
- `alpha` (`opytimize.optimizer.swarm.sbo.SBO` property), 117
- `AO` (class in `opytimize.optimizer.population.ao`), 51
- `AOA` (class in `opytimize.optimizer.misc.aoa`), 44
- `AP` (`opytimize.optimizer.swarm.csa.CSA` property), 98
- `area_limit` (`opytimize.optimizer.evolutionary.foa.FOA` property), 33
- `ArgumentError`, 134
- `ASO` (class in `opytimize.optimizer.science.aso`), 65
- B**
- `b` (`opytimize.optimizer.swarm.mfo.MFO` property), 111
- `b` (`opytimize.optimizer.swarm.woa.WOA` property), 123
- `BA` (class in `opytimize.optimizer.swarm.ba`), 93
- `best_agent` (`opytimize.core.space.Space` property), 12
- `best_position` (`opytimize.optimizer.population.loa.Lion` property), 57
- `best_tree` (`opytimize.spaces.tree.TreeSpace` property), 128
- `beta` (`opytimize.optimizer.population.osa.OSA` property), 61
- `beta` (`opytimize.optimizer.science.aig.AIG` property), 65
- `beta` (`opytimize.optimizer.science.aso.ASO` property), 66
- `beta` (`opytimize.optimizer.science.hgso.HGSO` property), 72
- `beta` (`opytimize.optimizer.science.sa.SA` property), 76
- `beta` (`opytimize.optimizer.science.two.TWO` property), 78
- `beta` (`opytimize.optimizer.science.ww0.WWO` property), 82
- `beta` (`opytimize.optimizer.swarm.cs.CS` property), 97
- `beta` (`opytimize.optimizer.swarm.eho.EHO` property), 99
- `beta` (`opytimize.optimizer.swarm.fa.FA` property), 100
- `beta` (`opytimize.optimizer.swarm.fpa.FPA` property), 101
- `beta` (`opytimize.optimizer.swarm.fso.FSO` property), 102
- `beta` (`opytimize.optimizer.swarm.js.JS` property), 104
- `BH` (class in `opytimize.optimizer.science.bh`), 67
- `Block` (class in `opytimize.core.block`), 6
- `BMRFO` (class in `opytimize.optimizer.boolean.bmrfo`), 25
- `BOA` (class in `opytimize.optimizer.swarm.boa`), 94
- `BooleanSpace` (class in `opytimize.spaces.boolean`), 125
- `bout_size` (`opytimize.optimizer.evolutionary.ep.EP` property), 31
- `BPSO` (class in `opytimize.optimizer.boolean.bps0`), 26
- `BSA` (class in `opytimize.optimizer.evolutionary.bsa`), 28
- `BSO` (class in `opytimize.optimizer.social.bso`), 83
- `build()` (`opytimize.core.optimizer.Optimizer` method), 11
- `build()` (`opytimize.core.space.Space` method), 13
- `build()` (`opytimize.spaces.pareto.ParetoSpace` method), 127
- `BuildError`, 134
- `built` (`opytimize.core.function.Function` property), 8
- `built` (`opytimize.core.optimizer.Optimizer` property), 11
- `built` (`opytimize.core.space.Space` property), 12
- `bw` (`opytimize.optimizer.evolutionary.hs.HS` property), 38
- `bw_max` (`opytimize.optimizer.evolutionary.hs.IHS` property), 38
- `bw_max` (`opytimize.optimizer.evolutionary.hs.SGHS` property), 40
- `bw_min` (`opytimize.optimizer.evolutionary.hs.IHS` property), 38
- `bw_min` (`opytimize.optimizer.evolutionary.hs.SGHS` property), 38

- property), 40
- BWO (class in `opyoptimizer.optimizers.swarm.bwo`), 95
- ## C
- C (`opyoptimizer.optimizers.science.eo.EO` property), 69
- c (`opyoptimizer.optimizers.science.wdo.WDO` property), 80
- c (`opyoptimizer.optimizers.social.ssd.SSD` property), 89
- c (`opyoptimizer.optimizers.swarm.boa.BOA` property), 94
- c1 (`opyoptimizer.optimizers.boolean.bpsb.BPSO` property), 27
- c1 (`opyoptimizer.optimizers.science.teo.TEO` property), 77
- c1 (`opyoptimizer.optimizers.swarm.af.AF` property), 92
- c1 (`opyoptimizer.optimizers.swarm.pso.PSO` property), 114
- c2 (`opyoptimizer.optimizers.boolean.bpsb.BPSO` property), 27
- c2 (`opyoptimizer.optimizers.science.teo.TEO` property), 77
- c2 (`opyoptimizer.optimizers.swarm.af.AF` property), 93
- c2 (`opyoptimizer.optimizers.swarm.pso.PSO` property), 114
- C_g (`opyoptimizer.optimizers.swarm.sso.SSO` property), 122
- c_max (`opyoptimizer.optimizers.swarm.goa.GOA` property), 103
- c_min (`opyoptimizer.optimizers.swarm.goa.GOA` property), 103
- C_p (`opyoptimizer.optimizers.swarm.sso.SSO` property), 122
- C_t (`opyoptimizer.optimizers.swarm.kh.KH` property), 106
- C_w (`opyoptimizer.optimizers.swarm.sso.SSO` property), 122
- Callback (class in `opyoptimizer.utils.callback`), 131
- callbacks (`opyoptimizer.utils.callback.CallbackVessel` property), 132
- CallbackVessel (class in `opyoptimizer.utils.callback`), 132
- category (`opyoptimizer.core.node.Node` property), 9
- Cell (class in `opyoptimizer.core.cell`), 7
- CEM (class in `opyoptimizer.optimizers.misc.cem`), 45
- Cf (`opyoptimizer.optimizers.swarm.stoa.STOA` property), 123
- chaotic_map (`opyoptimizer.optimizers.misc.doa.DOA` property), 47
- CheckpointCallback (class in `opyoptimizer.utils.callback`), 133
- child_ratio (`opyoptimizer.optimizers.evolutionary.es.ES` property), 32
- CI (class in `opyoptimizer.optimizers.social.ci`), 84
- clip_by_bound() (`opyoptimizer.core.agent.Agent` method), 6
- clip_by_bound() (`opyoptimizer.core.space.Space` method), 13
- clip_by_bound() (`opyoptimizer.spaces.pareto.ParetoSpace` method), 127
- clip_ratio (`opyoptimizer.optimizers.evolutionary.ep.EP` property), 31
- COA (class in `opyoptimizer.optimizers.population.coa`), 52
- coefficient (`opyoptimizer.optimizers.science.hgso.HGSO` property), 72
- compile() (`opyoptimizer.core.optimizer.Optimizer` method), 11
- compile() (`opyoptimizer.optimizers.boolean.bpsb.BPSO` method), 27
- compile() (`opyoptimizer.optimizers.evolutionary.bsa.BSA` method), 29
- compile() (`opyoptimizer.optimizers.evolutionary.ep.EP` method), 31
- compile() (`opyoptimizer.optimizers.evolutionary.es.ES` method), 32
- compile() (`opyoptimizer.optimizers.evolutionary.foa.FOA` method), 33
- compile() (`opyoptimizer.optimizers.evolutionary.hs.SGHS` method), 40
- compile() (`opyoptimizer.optimizers.misc.cem.CEM` method), 45
- compile() (`opyoptimizer.optimizers.misc.doa.DOA` method), 47
- compile() (`opyoptimizer.optimizers.misc.nds.NDS` method), 48
- compile() (`opyoptimizer.optimizers.population.coa.COA` method), 52
- compile() (`opyoptimizer.optimizers.population.gco.GCO` method), 54
- compile() (`opyoptimizer.optimizers.population.loa.LOA` method), 58
- compile() (`opyoptimizer.optimizers.population.ppa.PPA` method), 62
- compile() (`opyoptimizer.optimizers.population.rfo.RFO` method), 64
- compile() (`opyoptimizer.optimizers.science.aso.ASO` method), 66
- compile() (`opyoptimizer.optimizers.science.eo.EO` method), 69
- compile() (`opyoptimizer.optimizers.science.esa.ESA` method), 70
- compile() (`opyoptimizer.optimizers.science.gsa.GSA` method), 71
- compile() (`opyoptimizer.optimizers.science.hgso.HGSO` method), 72
- compile() (`opyoptimizer.optimizers.science.lsa.LSA` method), 74
- compile() (`opyoptimizer.optimizers.science.moa.MOA` method), 75
- compile() (`opyoptimizer.optimizers.science.teo.TEO` method), 77
- compile() (`opyoptimizer.optimizers.science.wca.WCA` method), 79
- compile() (`opyoptimizer.optimizers.science.wdo.WDO` method), 80
- compile() (`opyoptimizer.optimizers.science.wwc.WWC` method), 82

- `compile()` (*opoptimizer.optimizers.social.ci.CI* method), 85
 - `compile()` (*opoptimizer.optimizers.social.isa.ISA* method), 86
 - `compile()` (*opoptimizer.optimizers.social.mvpa.MVPA* method), 86
 - `compile()` (*opoptimizer.optimizers.social.ssd.SSD* method), 89
 - `compile()` (*opoptimizer.optimizers.swarm.abc.ABC* method), 90
 - `compile()` (*opoptimizer.optimizers.swarm.af.AF* method), 93
 - `compile()` (*opoptimizer.optimizers.swarm.ba.BA* method), 94
 - `compile()` (*opoptimizer.optimizers.swarm.boa.BOA* method), 94
 - `compile()` (*opoptimizer.optimizers.swarm.csa.CSA* method), 98
 - `compile()` (*opoptimizer.optimizers.swarm.eho.EHO* method), 99
 - `compile()` (*opoptimizer.optimizers.swarm.ffoa.FFOA* method), 101
 - `compile()` (*opoptimizer.optimizers.swarm.js.JS* method), 104
 - `compile()` (*opoptimizer.optimizers.swarm.kh.KH* method), 107
 - `compile()` (*opoptimizer.optimizers.swarm.pio.PIO* method), 113
 - `compile()` (*opoptimizer.optimizers.swarm.pso.PSO* method), 114
 - `compile()` (*opoptimizer.optimizers.swarm.pso.RPSO* method), 116
 - `compile()` (*opoptimizer.optimizers.swarm.pso.VPSO* method), 117
 - `compile()` (*opoptimizer.optimizers.swarm.sbo.SBO* method), 117
 - `compile()` (*opoptimizer.optimizers.swarm.sfo.SFO* method), 119
 - `compile()` (*opoptimizer.optimizers.swarm.sso.SSO* method), 122
 - `constant` (*opoptimizer.optimizers.science.hgso.HGSO* property), 72
 - `ConstrainedFunction` (class in *opoptimizer.functions.constrained*), 15
 - `constraints` (*opoptimizer.functions.constrained.ConstrainedFunction* property), 15
 - `count` (*opoptimizer.optimizers.misc.nds.NDS* property), 48
 - `counter` (*opoptimizer.optimizers.population.gco.GCO* property), 54
 - `CR` (*opoptimizer.optimizers.evolutionary.de.DE* property), 30
 - `CR` (*opoptimizer.optimizers.population.gco.GCO* property), 54
 - `cr` (*opoptimizer.optimizers.swarm.bwo.BWO* property), 96
 - `Cr` (*opoptimizer.optimizers.swarm.kh.KH* property), 106
 - `CS` (class in *opoptimizer.optimizers.swarm.cs*), 96
 - `CSA` (class in *opoptimizer.optimizers.swarm.csa*), 98
- ## D
- `D` (*opoptimizer.optimizers.science.esa.ESA* property), 70
 - `d_max` (*opoptimizer.optimizers.science.wca.WCA* property), 79
 - `D_max` (*opoptimizer.optimizers.swarm.kh.KH* property), 106
 - `d_root` (*opoptimizer.optimizers.evolutionary.rra.RRA* property), 43
 - `d_runner` (*opoptimizer.optimizers.evolutionary.rra.RRA* property), 43
 - `DE` (class in *opoptimizer.optimizers.evolutionary.de*), 29
 - `decay` (*opoptimizer.optimizers.social.ssd.SSD* property), 89
 - `delta` (*opoptimizer.optimizers.population.ao.AO* property), 51
 - `delta_t` (*opoptimizer.optimizers.science.two.TWO* property), 78
 - `direction` (*opoptimizer.optimizers.science.lsa.LSA* property), 74
 - `DiscreteSearchCallback` (class in *opoptimizer.utils.callback*), 133
 - `DOA` (class in *opoptimizer.optimizers.misc.doa*), 46
 - `dump()` (*opoptimizer.utils.history.History* method), 135
- ## E
- `e` (*opoptimizer.optimizers.evolutionary.iwo.IWO* property), 42
 - `E` (*opoptimizer.optimizers.science.lsa.LSA* property), 73
 - `e` (*opoptimizer.optimizers.swarm.sfo.SFO* property), 119
 - `E_max` (*opoptimizer.optimizers.science.weo.WEO* property), 81
 - `E_min` (*opoptimizer.optimizers.science.weo.WEO* property), 81
 - `EFO` (class in *opoptimizer.optimizers.science.efo*), 68
 - `EHO` (class in *opoptimizer.optimizers.swarm.eho*), 99
 - `environment` (*opoptimizer.optimizers.science.teo.TEO* property), 77
 - `EO` (class in *opoptimizer.optimizers.science.eo*), 69
 - `EP` (class in *opoptimizer.optimizers.evolutionary.ep*), 30
 - `EPO` (class in *opoptimizer.optimizers.population.epo*), 53
 - `Error`, 134
 - `ES` (class in *opoptimizer.optimizers.evolutionary.es*), 32
 - `ESA` (class in *opoptimizer.optimizers.science.esa*), 70
 - `eta` (*opoptimizer.optimizers.swarm.fpa.FPA* property), 101
 - `eta` (*opoptimizer.optimizers.swarm.js.JS* property), 104
 - `euclidean_distance()` (in module *opoptimizer.math.general*), 20
 - `evaluate()` (*opoptimizer.core.optimizer.Optimizer* method), 11

`evaluate()` (*opytizer.optimizers.boolean.bps.BPSO method*), 27
`evaluate()` (*opytizer.optimizers.evolutionary.gp.GP method*), 37
`evaluate()` (*opytizer.optimizers.social.isa.ISA method*), 86
`evaluate()` (*opytizer.optimizers.social.ssd.SSD method*), 90
`evaluate()` (*opytizer.optimizers.swarm.csa.CSA method*), 98
`evaluate()` (*opytizer.optimizers.swarm.pso.PSO method*), 114
`evaluate()` (*opytizer.optimizers.swarm.sso.SSO method*), 122
`evaluate()` (*opytizer.Opytizer method*), 4
`evaluate_args` (*opytizer.Opytizer property*), 3

F

`F` (*opytizer.optimizers.evolutionary.bsa.BSA property*), 29
`F` (*opytizer.optimizers.evolutionary.de.DE property*), 30
`f` (*opytizer.optimizers.population.epo.EPO property*), 53
`F` (*opytizer.optimizers.population.gco.GCO property*), 54
`f` (*opytizer.optimizers.swarm.goa.GOA property*), 103
`f_max` (*opytizer.optimizers.swarm.ba.BA property*), 93
`f_min` (*opytizer.optimizers.swarm.ba.BA property*), 93
`FA` (*class in opytizer.optimizers.swarm.fa*), 100
`female` (*opytizer.optimizers.population.loa.Lion property*), 57
`FFOA` (*class in opytizer.optimizers.swarm.ffa*), 100
`file_path` (*opytizer.utils.callback.CheckpointCallback property*), 133
`fill_with_binary()` (*opytizer.core.agent.Agent method*), 6
`fill_with_static()` (*opytizer.core.agent.Agent method*), 6
`fill_with_uniform()` (*opytizer.core.agent.Agent method*), 6
`final_sigma` (*opytizer.optimizers.evolutionary.iwo.IWO property*), 42
`find_node()` (*opytizer.core.node.Node method*), 10
`fit` (*opytizer.core.agent.Agent property*), 5
`fitness` (*opytizer.optimizers.swarm.pso.AIWPSO property*), 115
`fl` (*opytizer.optimizers.swarm.csa.CSA property*), 98
`flag` (*opytizer.core.node.Node property*), 9
`flows` (*opytizer.optimizers.science.wca.WCA property*), 79
`FOA` (*class in opytizer.optimizers.evolutionary.foa*), 33
`foraging` (*opytizer.optimizers.swarm.kh.KH property*), 107

`FPA` (*class in opytizer.optimizers.swarm.fpa*), 101
`fragrance` (*opytizer.optimizers.swarm.boa.BOA property*), 94
`frequency` (*opytizer.optimizers.swarm.ba.BA property*), 93
`frequency` (*opytizer.utils.callback.CheckpointCallback property*), 133
`FSO` (*class in opytizer.optimizers.swarm.fso*), 102
`Function` (*class in opytizer.core.function*), 8
`function` (*opytizer.Opytizer property*), 3
`functions` (*opytizer.functions.multi_objective.standard.MultiObjectiveF property*), 16
`functions` (*opytizer.spaces.tree.TreeSpace property*), 128

G

`G` (*opytizer.optimizers.science.gsa.GSA property*), 71
`g` (*opytizer.optimizers.science.wdo.WDO property*), 80
`GA` (*class in opytizer.optimizers.evolutionary.ga*), 34
`gamma` (*opytizer.optimizers.swarm.fa.FA property*), 100
`gamma` (*opytizer.optimizers.swarm.js.JS property*), 104
`GCO` (*class in opytizer.optimizers.population.gco*), 53
`generate_bernoulli_distribution()` (*in module opytizer.math.distribution*), 19
`generate_binary_random_number()` (*in module opytizer.math.random*), 21
`generate_choice_distribution()` (*in module opytizer.math.distribution*), 19
`generate_exponential_random_number()` (*in module opytizer.math.random*), 21
`generate_gamma_random_number()` (*in module opytizer.math.random*), 22
`generate_gaussian_random_number()` (*in module opytizer.math.random*), 22
`generate_integer_random_number()` (*in module opytizer.math.random*), 22
`generate_levy_distribution()` (*in module opytizer.math.distribution*), 19
`generate_uniform_random_number()` (*in module opytizer.math.random*), 22
`get_console_handler()` (*in module opytizer.utils.logging*), 136
`get_convergence()` (*opytizer.utils.history.History method*), 135
`get_logger()` (*in module opytizer.utils.logging*), 136
`get_timed_file_handler()` (*in module opytizer.utils.logging*), 136
`GHS` (*class in opytizer.optimizers.evolutionary.hs*), 39
`GOA` (*class in opytizer.optimizers.swarm.goa*), 103
`GOGHS` (*class in opytizer.optimizers.evolutionary.hs*), 41
`GP` (*class in opytizer.optimizers.evolutionary.gp*), 35
`GP` (*opytizer.optimizers.science.eo.EO property*), 69
`grid` (*opytizer.spaces.grid.GridSpace property*), 126
`GridSpace` (*class in opytizer.spaces.grid*), 125

group (*opytizer.optimizers.population.loa.Lion* property), 57

grow() (*opytizer.spaces.tree.TreeSpace* method), 128

GS (*class in opytizer.optimizers.misc.gs*), 47

GSA (*class in opytizer.optimizers.science.gsa*), 71

GSC (*opytizer.optimizers.evolutionary.foa.FOA* property), 33

GWO (*class in opytizer.optimizers.population.gwo*), 55

H

h_max (*opytizer.optimizers.science.wwo.WWO* property), 82

HC (*class in opytizer.optimizers.misc.hc*), 47

height (*opytizer.optimizers.science.wwo.WWO* property), 82

HGSO (*class in opytizer.optimizers.science.hgso*), 72

HHO (*class in opytizer.optimizers.population.hho*), 55

History (*class in opytizer.utils.history*), 135

history (*opytizer.Opytizer* property), 3

HMCR (*opytizer.optimizers.evolutionary.hs.HS* property), 37

HMCR (*opytizer.optimizers.evolutionary.hs.SGHS* property), 39

HMCR_history (*opytizer.optimizers.evolutionary.hs.SGHS* property), 40

HMCRm (*opytizer.optimizers.evolutionary.hs.SGHS* property), 39

HS (*class in opytizer.optimizers.evolutionary.hs*), 37

HyperComplexSpace (*class in opytizer.spaces.hyper_complex*), 126

I

I (*opytizer.optimizers.population.loa.LOA* property), 58

IHS (*class in opytizer.optimizers.evolutionary.hs*), 38

init_sigma (*opytizer.optimizers.evolutionary.iwo.IWO* property), 42

InnerBlock (*class in opytizer.core.block*), 7

input_idx (*opytizer.core.cell.Cell* property), 8

InputBlock (*class in opytizer.core.block*), 7

ISA (*class in opytizer.optimizers.social.isa*), 85

iteration (*opytizer.Opytizer* property), 3

IWO (*class in opytizer.optimizers.evolutionary.iwo*), 42

J

JS (*class in opytizer.optimizers.swarm.js*), 104

K

K (*opytizer.optimizers.science.hgso.HGSO* property), 72

k (*opytizer.optimizers.social.bso.BSO* property), 84

k_max (*opytizer.optimizers.science.wwo.WWO* property), 82

KH (*class in opytizer.optimizers.swarm.kh*), 106

kmeans() (*in module opytizer.math.general*), 20

L

l (*opytizer.optimizers.population.epo.EPO* property), 53

l (*opytizer.optimizers.swarm.goa.GOA* property), 103

l1 (*opytizer.optimizers.science.hgso.HGSO* property), 72

l2 (*opytizer.optimizers.science.hgso.HGSO* property), 72

l3 (*opytizer.optimizers.science.hgso.HGSO* property), 72

last_best_fit (*opytizer.optimizers.evolutionary.rra.RRA* property), 43

lb (*opytizer.core.agent.Agent* property), 5

lb (*opytizer.core.space.Space* property), 12

left (*opytizer.core.node.Node* property), 9

length (*opytizer.optimizers.science.wwo.WWO* property), 82

life (*opytizer.optimizers.population.gco.GCO* property), 54

life_time (*opytizer.optimizers.evolutionary.foa.FOA* property), 33

Lion (*class in opytizer.optimizers.population.loa*), 57

LOA (*class in opytizer.optimizers.population.loa*), 57

load() (*opytizer.Opytizer* class method), 4

local_position (*opytizer.optimizers.boolean.bps.BPSO* property), 27

local_position (*opytizer.optimizers.social.isa.ISA* property), 86

local_position (*opytizer.optimizers.social.ssd.SSD* property), 89

local_position (*opytizer.optimizers.swarm.pso.PSO* property), 114

local_position (*opytizer.optimizers.swarm.sso.SSO* property), 122

Logger (*class in opytizer.utils.logging*), 135

loudness (*opytizer.optimizers.swarm.ba.BA* property), 94

lower (*opytizer.optimizers.social.ci.CI* property), 85

lower_bound (*opytizer.optimizers.boolean.umd.UMDA* property), 28

LP (*opytizer.optimizers.evolutionary.hs.SGHS* property), 39

lp (*opytizer.optimizers.evolutionary.hs.SGHS* property), 40

LSA (*class in opytizer.optimizers.science.lsa*), 73

LSC (*opytizer.optimizers.evolutionary.foa.FOA* property), 33

M

- `m` (*opytimizer.optimizers.social.bso.BSO* property), 83
- `m` (*opytimizer.optimizers.swarm.af.AF* property), 93
- `Ma` (*opytimizer.optimizers.population.loa.LOA* property), 58
- `mapped_position` (*opytimizer.core.agent.Agent* property), 6
- `mapping` (*opytimizer.core.agent.Agent* property), 6
- `mapping` (*opytimizer.core.space.Space* property), 12
- `mass` (*opytimizer.optimizers.swarm.pso.RPSO* property), 116
- `max_depth` (*opytimizer.core.node.Node* property), 9
- `max_depth` (*opytimizer.spaces.tree.TreeSpace* property), 128
- `max_seeds` (*opytimizer.optimizers.evolutionary.iwo.IWO* property), 42
- `max_stall` (*opytimizer.optimizers.evolutionary.rra.RRA* property), 43
- `max_time` (*opytimizer.optimizers.science.lsa.LSA* property), 73
- `mean` (*opytimizer.optimizers.misc.cem.CEM* property), 45
- `memory` (*opytimizer.optimizers.swarm.csa.CSA* property), 98
- `MFO` (class in *opytimizer.optimizers.swarm.mfo*), 111
- `min_depth` (*opytimizer.core.node.Node* property), 9
- `min_depth` (*opytimizer.spaces.tree.TreeSpace* property), 128
- `min_seeds` (*opytimizer.optimizers.evolutionary.iwo.IWO* property), 42
- `mix_rate` (*opytimizer.optimizers.evolutionary.bsa.BSA* property), 29
- `MOA` (class in *opytimizer.optimizers.science.moa*), 74
- `module`
 - `opytimizer.core`, 13
 - `opytimizer.core.agent`, 5
 - `opytimizer.core.block`, 6
 - `opytimizer.core.cell`, 7
 - `opytimizer.core.function`, 8
 - `opytimizer.core.node`, 9
 - `opytimizer.core.optimizer`, 11
 - `opytimizer.core.space`, 12
 - `opytimizer.functions`, 17
 - `opytimizer.functions.constrained`, 15
 - `opytimizer.functions.multi_objective`, 17
 - `opytimizer.functions.multi_objective.standard`, 16
 - `opytimizer.functions.multi_objective.weighted`, 16
 - `opytimizer.math`, 23
 - `opytimizer.math.distribution`, 19
 - `opytimizer.math.general`, 20
 - `opytimizer.math.random`, 21
 - `opytimizer.optimizers`, 124
 - `opytimizer.optimizers.boolean`, 28
 - `opytimizer.optimizers.boolean.bmrfo`, 25
 - `opytimizer.optimizers.boolean.bpso`, 26
 - `opytimizer.optimizers.boolean.umda`, 27
 - `opytimizer.optimizers.evolutionary`, 44
 - `opytimizer.optimizers.evolutionary.bsa`, 28
 - `opytimizer.optimizers.evolutionary.de`, 29
 - `opytimizer.optimizers.evolutionary.ep`, 30
 - `opytimizer.optimizers.evolutionary.es`, 32
 - `opytimizer.optimizers.evolutionary.foa`, 33
 - `opytimizer.optimizers.evolutionary.ga`, 34
 - `opytimizer.optimizers.evolutionary.gp`, 35
 - `opytimizer.optimizers.evolutionary.hs`, 37
 - `opytimizer.optimizers.evolutionary.iwo`, 42
 - `opytimizer.optimizers.evolutionary.rra`, 43
 - `opytimizer.optimizers.misc`, 49
 - `opytimizer.optimizers.misc.aoa`, 44
 - `opytimizer.optimizers.misc.cem`, 45
 - `opytimizer.optimizers.misc.doa`, 46
 - `opytimizer.optimizers.misc.gs`, 47
 - `opytimizer.optimizers.misc.hc`, 47
 - `opytimizer.optimizers.misc.nds`, 48
 - `opytimizer.optimizers.population`, 65
 - `opytimizer.optimizers.population.aeo`, 49
 - `opytimizer.optimizers.population.ao`, 51
 - `opytimizer.optimizers.population.coa`, 52
 - `opytimizer.optimizers.population.epo`, 53
 - `opytimizer.optimizers.population.gco`, 53
 - `opytimizer.optimizers.population.gwo`, 55
 - `opytimizer.optimizers.population.hho`, 55
 - `opytimizer.optimizers.population.loa`, 57
 - `opytimizer.optimizers.population.osa`, 61
 - `opytimizer.optimizers.population.ppa`, 62
 - `opytimizer.optimizers.population.pvs`, 63
 - `opytimizer.optimizers.population.rfo`, 64
 - `opytimizer.optimizers.science`, 83
 - `opytimizer.optimizers.science.aig`, 65
 - `opytimizer.optimizers.science.aso`, 65
 - `opytimizer.optimizers.science.bh`, 67
 - `opytimizer.optimizers.science.efo`, 68
 - `opytimizer.optimizers.science.eo`, 69
 - `opytimizer.optimizers.science.esa`, 70
 - `opytimizer.optimizers.science.gsa`, 71
 - `opytimizer.optimizers.science.hgso`, 72
 - `opytimizer.optimizers.science.lsa`, 73
 - `opytimizer.optimizers.science.moa`, 74
 - `opytimizer.optimizers.science.mvo`, 75
 - `opytimizer.optimizers.science.sa`, 76
 - `opytimizer.optimizers.science.teo`, 76
 - `opytimizer.optimizers.science.two`, 77
 - `opytimizer.optimizers.science.wca`, 78

opytimizer.optimizers.science.wdo, 80
 opytimizer.optimizers.science.weo, 80
 opytimizer.optimizers.science.woo, 81
 opytimizer.optimizers.social, 90
 opytimizer.optimizers.social.bso, 83
 opytimizer.optimizers.social.ci, 84
 opytimizer.optimizers.social.isa, 85
 opytimizer.optimizers.social.mvpa, 86
 opytimizer.optimizers.social.qsa, 87
 opytimizer.optimizers.social.ssd, 88
 opytimizer.optimizers.swarm, 124
 opytimizer.optimizers.swarm.abc, 90
 opytimizer.optimizers.swarm.abo, 91
 opytimizer.optimizers.swarm.af, 92
 opytimizer.optimizers.swarm.ba, 93
 opytimizer.optimizers.swarm.boa, 94
 opytimizer.optimizers.swarm.bwo, 95
 opytimizer.optimizers.swarm.cs, 96
 opytimizer.optimizers.swarm.csa, 98
 opytimizer.optimizers.swarm.eho, 99
 opytimizer.optimizers.swarm.fa, 100
 opytimizer.optimizers.swarm.ffoa, 100
 opytimizer.optimizers.swarm.fpa, 101
 opytimizer.optimizers.swarm.fso, 102
 opytimizer.optimizers.swarm.goa, 103
 opytimizer.optimizers.swarm.js, 104
 opytimizer.optimizers.swarm.kh, 106
 opytimizer.optimizers.swarm.mfo, 111
 opytimizer.optimizers.swarm.mrfo, 111
 opytimizer.optimizers.swarm.pio, 113
 opytimizer.optimizers.swarm.pso, 114
 opytimizer.optimizers.swarm.sbo, 117
 opytimizer.optimizers.swarm.sca, 118
 opytimizer.optimizers.swarm.sfo, 119
 opytimizer.optimizers.swarm.sos, 120
 opytimizer.optimizers.swarm.ssa, 121
 opytimizer.optimizers.swarm.sso, 122
 opytimizer.optimizers.swarm.stoa, 123
 opytimizer.optimizers.swarm.woa, 123
 opytimizer.spaces, 129
 opytimizer.spaces.boolean, 125
 opytimizer.spaces.graph, 125
 opytimizer.spaces.grid, 125
 opytimizer.spaces.hyper_complex, 126
 opytimizer.spaces.pareto, 126
 opytimizer.spaces.search, 127
 opytimizer.spaces.tree, 127
 opytimizer.utils, 136
 opytimizer.utils.callback, 131
 opytimizer.utils.constant, 133
 opytimizer.utils.exception, 134
 opytimizer.utils.history, 135
 opytimizer.utils.logging, 135
 opytimizer.visualization, 138

opytimizer.visualization.convergence, 137
 opytimizer.visualization.surface, 137
 motion (*opytimizer.optimizers.swarm.kh.KH* property), 107
 MRFO (*class in opytimizer.optimizers.swarm.mrfo*), 111
 mu (*opytimizer.optimizers.misc.aoa.AOA* property), 45
 Mu (*opytimizer.optimizers.population.loa.LOA* property), 58
 Mu (*opytimizer.optimizers.swarm.kh.KH* property), 106
 mu_k (*opytimizer.optimizers.science.two.TWO* property), 78
 mu_s (*opytimizer.optimizers.science.two.TWO* property), 78
 MultiObjectiveFunction (*class in opytimizer.functions.multi_objective.standard*), 16
 MultiObjectiveWeightedFunction (*class in opytimizer.functions.multi_objective.weighted*), 16
 MVO (*class in opytimizer.optimizers.science.mvo*), 75
 MVPA (*class in opytimizer.optimizers.social.mvpa*), 86

N

N (*opytimizer.optimizers.population.loa.LOA* property), 58
 n_agents (*opytimizer.core.space.Space* property), 12
 n_c (*opytimizer.optimizers.population.coa.COA* property), 52
 n_c1 (*opytimizer.optimizers.swarm.pio.PIO* property), 113
 n_c2 (*opytimizer.optimizers.swarm.pio.PIO* property), 113
 n_children (*opytimizer.optimizers.evolutionary.es.ES* property), 32
 n_ci (*opytimizer.optimizers.swarm.eho.EHO* property), 99
 n_clans (*opytimizer.optimizers.swarm.eho.EHO* property), 99
 n_clusters (*opytimizer.optimizers.science.hgso.HGSO* property), 72
 n_cycles (*opytimizer.optimizers.population.ao.AO* property), 51
 n_dimensions (*opytimizer.core.agent.Agent* property), 5
 n_dimensions (*opytimizer.core.space.Space* property), 12
 n_electrons (*opytimizer.optimizers.science.esa.ESA* property), 70
 n_input (*opytimizer.core.block.Block* property), 6
 n_leaves (*opytimizer.core.node.Node* property), 9
 N_max (*opytimizer.optimizers.swarm.kh.KH* property), 106
 n_nodes (*opytimizer.core.node.Node* property), 10
 n_output (*opytimizer.core.block.Block* property), 7
 n_p (*opytimizer.optimizers.population.coa.COA* property), 52

`n_p` (*opytimizer.optimizers.social.mvpa.MVPA* property), 86
`n_p` (*opytimizer.optimizers.swarm.pio.PIO* property), 113
`n_pareto_points` (*opytimizer.optimizers.misc.nds.NDS* property), 48
`n_replacement` (*opytimizer.optimizers.population.rfo.RFO* property), 64
`n_stall` (*opytimizer.optimizers.evolutionary.rra.RRA* property), 43
`n_teams` (*opytimizer.optimizers.social.mvpa.MVPA* property), 86
`n_terminals` (*opytimizer.spaces.tree.TreeSpace* property), 128
`n_TM` (*opytimizer.optimizers.science.teo.TEO* property), 77
`n_trials` (*opytimizer.optimizers.swarm.abc.ABC* property), 90
`n_updates` (*opytimizer.optimizers.misc.cem.CEM* property), 45
`n_variables` (*opytimizer.core.agent.Agent* property), 5
`n_variables` (*opytimizer.core.space.Space* property), 12
`n_wise()` (in module *opytimizer.math.general*), 20
`name` (*opytimizer.core.function.Function* property), 8
`name` (*opytimizer.core.node.Node* property), 9
`NBJS` (class in *opytimizer.optimizers.swarm.js*), 105
`NDS` (class in *opytimizer.optimizers.misc.nds*), 48
`negative_field` (*opytimizer.optimizers.science.efo.EFO* property), 68
`NGHS` (class in *opytimizer.optimizers.evolutionary.hs*), 40
`NN` (*opytimizer.optimizers.swarm.kh.KH* property), 106
`Node` (class in *opytimizer.core.node*), 9
`nomad` (*opytimizer.optimizers.population.loa.Lion* property), 57
`nsr` (*opytimizer.optimizers.science.wca.WCA* property), 79
O
`old_agents` (*opytimizer.optimizers.evolutionary.bsa.BSA* property), 29
`on_evaluate_after()` (*opytimizer.utils.callback.Callback* method), 131
`on_evaluate_after()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_evaluate_before()` (*opytimizer.utils.callback.Callback* method), 131
`on_evaluate_before()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_evaluate_before()` (*opytimizer.utils.callback.DiscreteSearchCallback* method), 133
`on_iteration_begin()` (*opytimizer.utils.callback.Callback* method), 131
`on_iteration_begin()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_iteration_end()` (*opytimizer.utils.callback.Callback* method), 131
`on_iteration_end()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_iteration_end()` (*opytimizer.utils.callback.CheckpointCallback* method), 133
`on_task_begin()` (*opytimizer.utils.callback.Callback* method), 131
`on_task_begin()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_task_begin()` (*opytimizer.utils.callback.DiscreteSearchCallback* method), 133
`on_task_end()` (*opytimizer.utils.callback.Callback* method), 131
`on_task_end()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_update_after()` (*opytimizer.utils.callback.Callback* method), 132
`on_update_after()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`on_update_before()` (*opytimizer.utils.callback.Callback* method), 132
`on_update_before()` (*opytimizer.utils.callback.CallbackVessel* method), 132
`Optimizer` (class in *opytimizer.core.optimizer*), 11
`optimizer` (*opytimizer.Opytimizer* property), 3
`Opytimizer` (class in *opytimizer*), 3
`opytimizer.core` module, 13
`opytimizer.core.agent` module, 5
`opytimizer.core.block` module, 6
`opytimizer.core.cell` module, 7
`opytimizer.core.function` module, 8
`opytimizer.core.node`

- module, 9
- opytimizer.core.optimizer
 - module, 11
- opytimizer.core.space
 - module, 12
- opytimizer.functions
 - module, 17
- opytimizer.functions.constrained
 - module, 15
- opytimizer.functions.multi_objective
 - module, 17
- opytimizer.functions.multi_objective.standard
 - module, 16
- opytimizer.functions.multi_objective.weighted
 - module, 16
- opytimizer.math
 - module, 23
- opytimizer.math.distribution
 - module, 19
- opytimizer.math.general
 - module, 20
- opytimizer.math.random
 - module, 21
- opytimizer.optimizers
 - module, 124
- opytimizer.optimizers.boolean
 - module, 28
- opytimizer.optimizers.boolean.bmrfo
 - module, 25
- opytimizer.optimizers.boolean.bpsa
 - module, 26
- opytimizer.optimizers.boolean.umd
 - module, 27
- opytimizer.optimizers.evolutionary
 - module, 44
- opytimizer.optimizers.evolutionary.bsa
 - module, 28
- opytimizer.optimizers.evolutionary.de
 - module, 29
- opytimizer.optimizers.evolutionary.ep
 - module, 30
- opytimizer.optimizers.evolutionary.es
 - module, 32
- opytimizer.optimizers.evolutionary.foa
 - module, 33
- opytimizer.optimizers.evolutionary.ga
 - module, 34
- opytimizer.optimizers.evolutionary.gp
 - module, 35
- opytimizer.optimizers.evolutionary.hs
 - module, 37
- opytimizer.optimizers.evolutionary.iwo
 - module, 42
- opytimizer.optimizers.evolutionary.rra
 - module, 43
- opytimizer.optimizers.misc
 - module, 49
- opytimizer.optimizers.misc.aoa
 - module, 44
- opytimizer.optimizers.misc.cem
 - module, 45
- opytimizer.optimizers.misc.doa
 - module, 46
- opytimizer.optimizers.misc.gs
 - module, 47
- opytimizer.optimizers.misc.hc
 - module, 47
- opytimizer.optimizers.misc.nds
 - module, 48
- opytimizer.optimizers.population
 - module, 65
- opytimizer.optimizers.population.aeo
 - module, 49
- opytimizer.optimizers.population.ao
 - module, 51
- opytimizer.optimizers.population.coa
 - module, 52
- opytimizer.optimizers.population.epo
 - module, 53
- opytimizer.optimizers.population.gco
 - module, 53
- opytimizer.optimizers.population.gwo
 - module, 55
- opytimizer.optimizers.population.hho
 - module, 55
- opytimizer.optimizers.population.loa
 - module, 57
- opytimizer.optimizers.population.osa
 - module, 61
- opytimizer.optimizers.population.ppa
 - module, 62
- opytimizer.optimizers.population.pvs
 - module, 63
- opytimizer.optimizers.population.rfo
 - module, 64
- opytimizer.optimizers.science
 - module, 83
- opytimizer.optimizers.science.aig
 - module, 65
- opytimizer.optimizers.science.aso
 - module, 65
- opytimizer.optimizers.science.bh
 - module, 67
- opytimizer.optimizers.science.efo
 - module, 68
- opytimizer.optimizers.science.eo
 - module, 69
- opytimizer.optimizers.science.esa

- module, 70
- opyoptimizer.optimizers.science.gsa
 - module, 71
- opyoptimizer.optimizers.science.hgso
 - module, 72
- opyoptimizer.optimizers.science.lsa
 - module, 73
- opyoptimizer.optimizers.science.moa
 - module, 74
- opyoptimizer.optimizers.science.mvo
 - module, 75
- opyoptimizer.optimizers.science.sa
 - module, 76
- opyoptimizer.optimizers.science.teo
 - module, 76
- opyoptimizer.optimizers.science.two
 - module, 77
- opyoptimizer.optimizers.science.wca
 - module, 78
- opyoptimizer.optimizers.science.wdo
 - module, 80
- opyoptimizer.optimizers.science.weo
 - module, 80
- opyoptimizer.optimizers.science.woa
 - module, 81
- opyoptimizer.optimizers.social
 - module, 90
- opyoptimizer.optimizers.social.bso
 - module, 83
- opyoptimizer.optimizers.social.ci
 - module, 84
- opyoptimizer.optimizers.social.isa
 - module, 85
- opyoptimizer.optimizers.social.mvpa
 - module, 86
- opyoptimizer.optimizers.social.qsa
 - module, 87
- opyoptimizer.optimizers.social.ssd
 - module, 88
- opyoptimizer.optimizers.swarm
 - module, 124
- opyoptimizer.optimizers.swarm.abc
 - module, 90
- opyoptimizer.optimizers.swarm.abo
 - module, 91
- opyoptimizer.optimizers.swarm.af
 - module, 92
- opyoptimizer.optimizers.swarm.ba
 - module, 93
- opyoptimizer.optimizers.swarm.boa
 - module, 94
- opyoptimizer.optimizers.swarm.bwo
 - module, 95
- opyoptimizer.optimizers.swarm.cs

- module, 96
- opyoptimizer.optimizers.swarm.csa
 - module, 98
- opyoptimizer.optimizers.swarm.eho
 - module, 99
- opyoptimizer.optimizers.swarm.fa
 - module, 100
- opyoptimizer.optimizers.swarm.ffa
 - module, 100
- opyoptimizer.optimizers.swarm.fpa
 - module, 101
- opyoptimizer.optimizers.swarm.fso
 - module, 102
- opyoptimizer.optimizers.swarm.goa
 - module, 103
- opyoptimizer.optimizers.swarm.js
 - module, 104
- opyoptimizer.optimizers.swarm.kh
 - module, 106
- opyoptimizer.optimizers.swarm.mfo
 - module, 111
- opyoptimizer.optimizers.swarm.mrfo
 - module, 111
- opyoptimizer.optimizers.swarm.pio
 - module, 113
- opyoptimizer.optimizers.swarm.pso
 - module, 114
- opyoptimizer.optimizers.swarm.sbo
 - module, 117
- opyoptimizer.optimizers.swarm.sca
 - module, 118
- opyoptimizer.optimizers.swarm.sfo
 - module, 119
- opyoptimizer.optimizers.swarm.sos
 - module, 120
- opyoptimizer.optimizers.swarm.ssa
 - module, 121
- opyoptimizer.optimizers.swarm.sso
 - module, 122
- opyoptimizer.optimizers.swarm.stoa
 - module, 123
- opyoptimizer.optimizers.swarm.woa
 - module, 123
- opyoptimizer.spaces
 - module, 129
- opyoptimizer.spaces.boolean
 - module, 125
- opyoptimizer.spaces.graph
 - module, 125
- opyoptimizer.spaces.grid
 - module, 125
- opyoptimizer.spaces.hyper_complex
 - module, 126
- opyoptimizer.spaces.pareto

module, 126
 opytimizer.spaces.search
 module, 127
 opytimizer.spaces.tree
 module, 127
 opytimizer.utils
 module, 136
 opytimizer.utils.callback
 module, 131
 opytimizer.utils.constant
 module, 133
 opytimizer.utils.exception
 module, 134
 opytimizer.utils.history
 module, 135
 opytimizer.utils.logging
 module, 135
 opytimizer.visualization
 module, 138
 opytimizer.visualization.convergence
 module, 137
 opytimizer.visualization.surface
 module, 137
 OSA (class in opytimizer.optimizers.population.osa), 61
 output_idx (opytimizer.core.cell.Cell property), 8
 OutputBlock (class in opytimizer.core.block), 7

P

P (opytimizer.optimizers.population.loa.LOA property), 58
 p (opytimizer.optimizers.science.mvo.MVO property), 75
 p (opytimizer.optimizers.swarm.boa.BOA property), 94
 p (opytimizer.optimizers.swarm.cs.CS property), 97
 p (opytimizer.optimizers.swarm.fpa.FPA property), 101
 p_crossover (opytimizer.optimizers.evolutionary.ga.GA property), 34
 p_crossover (opytimizer.optimizers.evolutionary.gp.GP property), 36
 p_double_best (opytimizer.optimizers.social.bso.BSO property), 84
 p_fit (opytimizer.optimizers.population.loa.Lion property), 57
 p_fork (opytimizer.optimizers.science.lsa.LSA property), 73
 p_mutation (opytimizer.optimizers.evolutionary.ga.GA property), 34
 p_mutation (opytimizer.optimizers.evolutionary.gp.GP property), 36
 p_mutation (opytimizer.optimizers.swarm.sbo.SBO property), 117
 p_replacement (opytimizer.optimizers.population.rfo.RFO property), 64
 p_replacement_cluster (opytimizer.optimizers.social.bso.BSO property), 84
 p_reproduction (opytimizer.optimizers.evolutionary.gp.GP property), 36
 p_selection (opytimizer.optimizers.boolean.umd.UMDA property), 27
 p_selection (opytimizer.optimizers.evolutionary.ga.GA property), 34
 p_single_best (opytimizer.optimizers.social.bso.BSO property), 84
 p_single_cluster (opytimizer.optimizers.social.bso.BSO property), 84
 PAR (opytimizer.optimizers.evolutionary.hs.HS property), 37
 PAR (opytimizer.optimizers.evolutionary.hs.SGHS property), 39
 PAR_history (opytimizer.optimizers.evolutionary.hs.SGHS property), 40
 PAR_max (opytimizer.optimizers.evolutionary.hs.IHS property), 38
 PAR_min (opytimizer.optimizers.evolutionary.hs.IHS property), 38
 params (opytimizer.core.optimizer.Optimizer property), 11
 parent (opytimizer.core.node.Node property), 9
 ParetoSpace (class in opytimizer.spaces.pareto), 126
 PARm (opytimizer.optimizers.evolutionary.hs.SGHS property), 39
 penalty (opytimizer.functions.constrained.ConstrainedFunction property), 15
 phi (opytimizer.optimizers.population.rfo.RFO property), 64
 phi (opytimizer.optimizers.science.efo.EFO property), 68
 PIO (class in opytimizer.optimizers.swarm.pio), 113
 plot() (in module opytimizer.visualization.convergence), 137
 plot() (in module opytimizer.visualization.surface), 137
 pm (opytimizer.optimizers.evolutionary.hs.NGHS property), 41
 pm (opytimizer.optimizers.swarm.bwo.BWO property), 96
 pointer (opytimizer.core.block.Block property), 6
 pointer (opytimizer.core.function.Function property), 8
 position (opytimizer.core.agent.Agent property), 5
 position (opytimizer.core.node.Node property), 10
 positive_field (opytimizer.optimizers.science.efo.EFO property), 68
 post_order (opytimizer.core.node.Node property), 10
 pp (opytimizer.optimizers.swarm.bwo.BWO property), 96
 PP (opytimizer.optimizers.swarm.sfo.SFO property), 119
 PPA (class in opytimizer.optimizers.population.ppa), 62
 pre_order (opytimizer.core.node.Node property), 10
 pressure (opytimizer.optimizers.science.hgso.HGSO

property), 72
 pride (*opyoptimizer.optimizers.population.loa.Lion* property), 57
 pro (*opyoptimizer.optimizers.science.teo.TEO* property), 77
 pruning_ratio (*opyoptimizer.optimizers.evolutionary.gp.GP* property), 36
 ps_ratio (*opyoptimizer.optimizers.science.efo.EFO* property), 68
 PSO (class in *opyoptimizer.optimizers.swarm.pso*), 114
 pulse_rate (*opyoptimizer.optimizers.swarm.ba.BA* property), 94
 PVS (class in *opyoptimizer.optimizers.population.pvs*), 63

Q

Q (*opyoptimizer.optimizers.swarm.af.AF* property), 93
 QSA (class in *opyoptimizer.optimizers.social.qsa*), 87

R

r (*opyoptimizer.optimizers.misc.doa.DOA* property), 46
 R (*opyoptimizer.optimizers.population.loa.LOA* property), 58
 r (*opyoptimizer.optimizers.social.ci.CI* property), 85
 r (*opyoptimizer.optimizers.swarm.ba.BA* property), 93
 R (*opyoptimizer.optimizers.swarm.pio.PIO* property), 113
 r_max (*opyoptimizer.optimizers.swarm.sca.SCA* property), 118
 r_mean (*opyoptimizer.optimizers.misc.hc.HC* property), 48
 r_min (*opyoptimizer.optimizers.swarm.sca.SCA* property), 118
 r_ratio (*opyoptimizer.optimizers.science.efo.EFO* property), 68
 r_var (*opyoptimizer.optimizers.misc.hc.HC* property), 48
 RFO (class in *opyoptimizer.optimizers.population.rfo*), 64
 rho (*opyoptimizer.optimizers.science.moa.MOA* property), 75
 RI (*opyoptimizer.optimizers.science.efo.EFO* property), 68
 right (*opyoptimizer.core.node.Node* property), 9
 RPSO (class in *opyoptimizer.optimizers.swarm.pso*), 115
 RRA (class in *opyoptimizer.optimizers.evolutionary.rra*), 43
 RT (*opyoptimizer.optimizers.science.wdo.WDO* property), 80

S

S (*opyoptimizer.optimizers.boolean.bmrfo.BMRFO* property), 25
 S (*opyoptimizer.optimizers.population.loa.LOA* property), 58
 S (*opyoptimizer.optimizers.swarm.mrfo.MRFO* property), 111
 SA (class in *opyoptimizer.optimizers.science.sa*), 76
 sardines (*opyoptimizer.optimizers.swarm.sfo.SFO* property), 119
 save() (*opyoptimizer.Opyoptimizer* method), 4

save_agents (*opyoptimizer.utils.history.History* property), 135
 SAVPSO (class in *opyoptimizer.optimizers.swarm.pso*), 116
 SBO (class in *opyoptimizer.optimizers.swarm.sbo*), 117
 SCA (class in *opyoptimizer.optimizers.swarm.sca*), 118
 SearchSpace (class in *opyoptimizer.spaces.search*), 127
 set (*opyoptimizer.optimizers.misc.nds.NDS* property), 48
 SFO (class in *opyoptimizer.optimizers.swarm.sfo*), 119
 SGHS (class in *opyoptimizer.optimizers.evolutionary.hs*), 39
 sigma (*opyoptimizer.optimizers.evolutionary.iwo.IWO* property), 42
 sigma (*opyoptimizer.optimizers.swarm.sbo.SBO* property), 117
 SizeError, 134
 SOS (class in *opyoptimizer.optimizers.swarm.sos*), 120
 Space (class in *opyoptimizer.core.space*), 12
 space (*opyoptimizer.Opyoptimizer* property), 3
 SSA (class in *opyoptimizer.optimizers.swarm.ssa*), 121
 SSD (class in *opyoptimizer.optimizers.social.ssd*), 88
 SSO (class in *opyoptimizer.optimizers.swarm.sso*), 122
 start() (*opyoptimizer.Opyoptimizer* method), 4
 status (*opyoptimizer.optimizers.misc.nds.NDS* property), 48
 std (*opyoptimizer.optimizers.misc.cem.CEM* property), 45
 step (*opyoptimizer.spaces.grid.GridSpace* property), 126
 STOA (class in *opyoptimizer.optimizers.swarm.stoa*), 123
 strategy (*opyoptimizer.optimizers.evolutionary.ep.EP* property), 31
 strategy (*opyoptimizer.optimizers.evolutionary.es.ES* property), 32
 sunspot_ratio (*opyoptimizer.optimizers.swarm.abo.ABO* property), 92

T

T (*opyoptimizer.optimizers.science.sa.SA* property), 76
 t (*opyoptimizer.optimizers.social.ci.CI* property), 85
 tau (*opyoptimizer.optimizers.social.isa.ISA* property), 85
 TEO (class in *opyoptimizer.optimizers.science.teo*), 76
 terminals (*opyoptimizer.spaces.tree.TreeSpace* property), 128
 theta (*opyoptimizer.optimizers.population.rfo.RFO* property), 64
 theta_max (*opyoptimizer.optimizers.science.weo.WEO* property), 81
 theta_min (*opyoptimizer.optimizers.science.weo.WEO* property), 81
 time (*opyoptimizer.optimizers.science.lsa.LSA* property), 74
 TM (*opyoptimizer.optimizers.science.teo.TEO* property), 77
 to_file() (*opyoptimizer.utils.logging.Logger* method), 135
 tol (*opyoptimizer.optimizers.evolutionary.rra.RRA* property), 43
 total_iterations (*opyoptimizer.Opyoptimizer* property), 3

[tournament_selection\(\)](#) (in module `optimizer.math.general`), 21
[transfer_rate](#) (`optimizer.optimizers.evolutionary.foa.FOA` property), 33
[trees](#) (`optimizer.spaces.tree.TreeSpace` property), 128
[TreeSpace](#) (class in `optimizer.spaces.tree`), 127
[trial](#) (`optimizer.optimizers.swarm.abc.ABC` property), 90
[ts](#) (`optimizer.core.agent.Agent` property), 6
[TWO](#) (class in `optimizer.optimizers.science.two`), 77
[type](#) (`optimizer.core.block.Block` property), 6
[TypeError](#), 134

U

[U](#) (`optimizer.optimizers.population.ao.AO` property), 51
[u](#) (`optimizer.optimizers.swarm.stoa.STOA` property), 123
[ub](#) (`optimizer.core.agent.Agent` property), 5
[ub](#) (`optimizer.core.space.Space` property), 12
[UMDA](#) (class in `optimizer.optimizers.boolean.umda`), 27
[update\(\)](#) (`optimizer.core.optimizer.Optimizer` method), 11
[update\(\)](#) (`optimizer.optimizers.boolean.bmrfo.BMRFO` method), 26
[update\(\)](#) (`optimizer.optimizers.boolean.bpsb.BPSB` method), 27
[update\(\)](#) (`optimizer.optimizers.boolean.umda.UMDA` method), 28
[update\(\)](#) (`optimizer.optimizers.evolutionary.bsa.BSA` method), 29
[update\(\)](#) (`optimizer.optimizers.evolutionary.de.DE` method), 30
[update\(\)](#) (`optimizer.optimizers.evolutionary.ep.EP` method), 31
[update\(\)](#) (`optimizer.optimizers.evolutionary.es.ES` method), 33
[update\(\)](#) (`optimizer.optimizers.evolutionary.foa.FOA` method), 34
[update\(\)](#) (`optimizer.optimizers.evolutionary.ga.GA` method), 35
[update\(\)](#) (`optimizer.optimizers.evolutionary.gp.GP` method), 37
[update\(\)](#) (`optimizer.optimizers.evolutionary.hs.GOGHS` method), 41
[update\(\)](#) (`optimizer.optimizers.evolutionary.hs.HS` method), 38
[update\(\)](#) (`optimizer.optimizers.evolutionary.hs.IHS` method), 38
[update\(\)](#) (`optimizer.optimizers.evolutionary.hs.NGHS` method), 41
[update\(\)](#) (`optimizer.optimizers.evolutionary.hs.SGHS` method), 40
[update\(\)](#) (`optimizer.optimizers.evolutionary.iwo.IWO` method), 43
[update\(\)](#) (`optimizer.optimizers.evolutionary.rra.RRA` method), 44
[update\(\)](#) (`optimizer.optimizers.misc.aoa.AOA` method), 45
[update\(\)](#) (`optimizer.optimizers.misc.cem.CEM` method), 46
[update\(\)](#) (`optimizer.optimizers.misc.doa.DOA` method), 47
[update\(\)](#) (`optimizer.optimizers.misc.hc.HC` method), 48
[update\(\)](#) (`optimizer.optimizers.misc.nds.NDS` method), 49
[update\(\)](#) (`optimizer.optimizers.population.aeo.AEO` method), 51
[update\(\)](#) (`optimizer.optimizers.population.ao.AO` method), 51
[update\(\)](#) (`optimizer.optimizers.population.coa.COA` method), 53
[update\(\)](#) (`optimizer.optimizers.population.epo.EPO` method), 53
[update\(\)](#) (`optimizer.optimizers.population.gco.GCO` method), 54
[update\(\)](#) (`optimizer.optimizers.population.gwo.GWO` method), 55
[update\(\)](#) (`optimizer.optimizers.population.hho.HHO` method), 56
[update\(\)](#) (`optimizer.optimizers.population.loa.LOA` method), 61
[update\(\)](#) (`optimizer.optimizers.population.osa.OSA` method), 61
[update\(\)](#) (`optimizer.optimizers.population.ppa.PPA` method), 63
[update\(\)](#) (`optimizer.optimizers.population.pvs.PVS` method), 63
[update\(\)](#) (`optimizer.optimizers.population.rfo.RFO` method), 65
[update\(\)](#) (`optimizer.optimizers.science.aig.AIG` method), 65
[update\(\)](#) (`optimizer.optimizers.science.aso.ASO` method), 67
[update\(\)](#) (`optimizer.optimizers.science.bh.BH` method), 68
[update\(\)](#) (`optimizer.optimizers.science.efo.EFO` method), 69
[update\(\)](#) (`optimizer.optimizers.science.eo.EO` method), 70
[update\(\)](#) (`optimizer.optimizers.science.esa.ESA` method), 70
[update\(\)](#) (`optimizer.optimizers.science.gsa.GSA` method), 71
[update\(\)](#) (`optimizer.optimizers.science.hgso.HGSO` method), 73
[update\(\)](#) (`optimizer.optimizers.science.lsa.LSA` method), 74

- `update()` (`opyoptimizer.optimizers.science.moa.MOA` method), 75
 - `update()` (`opyoptimizer.optimizers.science.mvo.MVO` method), 75
 - `update()` (`opyoptimizer.optimizers.science.sa.SA` method), 76
 - `update()` (`opyoptimizer.optimizers.science.teo.TEO` method), 77
 - `update()` (`opyoptimizer.optimizers.science.two.TWO` method), 78
 - `update()` (`opyoptimizer.optimizers.science.wca.WCA` method), 79
 - `update()` (`opyoptimizer.optimizers.science.wdo.WDO` method), 80
 - `update()` (`opyoptimizer.optimizers.science.weo.WEO` method), 81
 - `update()` (`opyoptimizer.optimizers.science.wwc.WWO` method), 83
 - `update()` (`opyoptimizer.optimizers.social.bso.BSO` method), 84
 - `update()` (`opyoptimizer.optimizers.social.ci.CI` method), 85
 - `update()` (`opyoptimizer.optimizers.social.isa.ISA` method), 86
 - `update()` (`opyoptimizer.optimizers.social.mvpa.MVPA` method), 87
 - `update()` (`opyoptimizer.optimizers.social.qsa.QSA` method), 88
 - `update()` (`opyoptimizer.optimizers.social.ssd.SSD` method), 90
 - `update()` (`opyoptimizer.optimizers.swarm.abc.ABC` method), 91
 - `update()` (`opyoptimizer.optimizers.swarm.abo.ABO` method), 92
 - `update()` (`opyoptimizer.optimizers.swarm.af.AF` method), 93
 - `update()` (`opyoptimizer.optimizers.swarm.ba.BA` method), 94
 - `update()` (`opyoptimizer.optimizers.swarm.boa.BOA` method), 95
 - `update()` (`opyoptimizer.optimizers.swarm.bwo.BWO` method), 96
 - `update()` (`opyoptimizer.optimizers.swarm.cs.CS` method), 97
 - `update()` (`opyoptimizer.optimizers.swarm.csa.CSA` method), 98
 - `update()` (`opyoptimizer.optimizers.swarm.eho.EHO` method), 100
 - `update()` (`opyoptimizer.optimizers.swarm.fa.FA` method), 100
 - `update()` (`opyoptimizer.optimizers.swarm.ffoa.FFOA` method), 101
 - `update()` (`opyoptimizer.optimizers.swarm.fpa.FPA` method), 102
 - `update()` (`opyoptimizer.optimizers.swarm.fso.FSO` method), 103
 - `update()` (`opyoptimizer.optimizers.swarm.goa.GOA` method), 103
 - `update()` (`opyoptimizer.optimizers.swarm.js.JS` method), 105
 - `update()` (`opyoptimizer.optimizers.swarm.kh.KH` method), 110
 - `update()` (`opyoptimizer.optimizers.swarm.mfo.MFO` method), 111
 - `update()` (`opyoptimizer.optimizers.swarm.mrfo.MRFO` method), 112
 - `update()` (`opyoptimizer.optimizers.swarm.pio.PIO` method), 114
 - `update()` (`opyoptimizer.optimizers.swarm.pso.AIWPSO` method), 115
 - `update()` (`opyoptimizer.optimizers.swarm.pso.PSO` method), 115
 - `update()` (`opyoptimizer.optimizers.swarm.pso.RPSO` method), 116
 - `update()` (`opyoptimizer.optimizers.swarm.pso.SAVPSO` method), 116
 - `update()` (`opyoptimizer.optimizers.swarm.pso.VPSO` method), 117
 - `update()` (`opyoptimizer.optimizers.swarm.sbo.SBO` method), 117
 - `update()` (`opyoptimizer.optimizers.swarm.sca.SCA` method), 118
 - `update()` (`opyoptimizer.optimizers.swarm.sfo.SFO` method), 120
 - `update()` (`opyoptimizer.optimizers.swarm.sos.SOS` method), 121
 - `update()` (`opyoptimizer.optimizers.swarm.ssa.SSA` method), 121
 - `update()` (`opyoptimizer.optimizers.swarm.sso.SSO` method), 122
 - `update()` (`opyoptimizer.optimizers.swarm.stoa.STOA` method), 123
 - `update()` (`opyoptimizer.optimizers.swarm.woa.WOA` method), 124
 - `update()` (`opyoptimizer.Opyoptimizer` method), 4
 - `update_args` (`opyoptimizer.Opyoptimizer` property), 3
 - `upper` (`opyoptimizer.optimizers.social.ci.CI` property), 85
 - `upper_bound` (`opyoptimizer.optimizers.boolean.umd.UMDA` property), 28
- ## V
- `V` (`opyoptimizer.optimizers.science.eo.EO` property), 69
 - `v` (`opyoptimizer.optimizers.swarm.stoa.STOA` property), 123
 - `V_f` (`opyoptimizer.optimizers.swarm.kh.KH` property), 106
 - `v_max` (`opyoptimizer.optimizers.science.wdo.WDO` property), 80
 - `v_velocity` (`opyoptimizer.optimizers.swarm.pso.VPSO` property), 117

`valid` (*opytimizer.core.cell.Cell* property), 8
`value` (*opytimizer.core.node.Node* property), 9
`ValueError`, 134
`velocity` (*opytimizer.optimizers.boolean.bps.BPSO* property), 27
`velocity` (*opytimizer.optimizers.population.ppa.PPA* property), 62
`velocity` (*opytimizer.optimizers.science.aso.ASO* property), 66
`velocity` (*opytimizer.optimizers.science.gsa.GSA* property), 71
`velocity` (*opytimizer.optimizers.science.wdo.WDO* property), 80
`velocity` (*opytimizer.optimizers.social.isa.ISA* property), 86
`velocity` (*opytimizer.optimizers.social.ssd.SSD* property), 89
`velocity` (*opytimizer.optimizers.swarm.ba.BA* property), 94
`velocity` (*opytimizer.optimizers.swarm.pio.PIO* property), 113
`velocity` (*opytimizer.optimizers.swarm.pso.PSO* property), 114
`VPSO` (class in *opytimizer.optimizers.swarm.pso*), 116

W

`w` (*opytimizer.optimizers.population.ao.AO* property), 51
`w` (*opytimizer.optimizers.social.isa.ISA* property), 85
`w` (*opytimizer.optimizers.swarm.pso.PSO* property), 114
`w_f` (*opytimizer.optimizers.swarm.kh.KH* property), 106
`w_max` (*opytimizer.optimizers.swarm.pso.AIWPSO* property), 115
`w_min` (*opytimizer.optimizers.swarm.pso.AIWPSO* property), 115
`w_n` (*opytimizer.optimizers.swarm.kh.KH* property), 106
`WCA` (class in *opytimizer.optimizers.science.wca*), 78
`WDO` (class in *opytimizer.optimizers.science.wdo*), 80
`weighted_wheel_selection()` (in module *opytimizer.math.general*), 21
`weights` (*opytimizer.functions.multi_objective.weighted.MultiObjectiveWeightedFunction* property), 16
`WEO` (class in *opytimizer.optimizers.science.weo*), 80
`WEP_max` (*opytimizer.optimizers.science.mvo.MVO* property), 75
`WEP_min` (*opytimizer.optimizers.science.mvo.MVO* property), 75
`WOA` (class in *opytimizer.optimizers.swarm.woa*), 123
`WWO` (class in *opytimizer.optimizers.science.wwo*), 81

X

`x_axis` (*opytimizer.optimizers.swarm.ffoa.FFOA* property), 101

Y

`y_axis` (*opytimizer.optimizers.swarm.ffoa.FFOA* property), 101

Z

`z` (*opytimizer.optimizers.swarm.sbo.SBO* property), 117