Opytimizer Documentation

Release 3.1.2

Gustavo de Rosa

Jan 13, 2023

PACKAGE REFERENCE

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+**.

CHAPTER

ONE

OPYTIMIZER

class opytimizer.**Opytimizer**(*space:* opytimizer.core.space.Space, *optimizer:* opytimizer.core.optimizer.Optimizer, *function:* opytimizer.core.function.Function, *save_agents: Optional[bool] = False*)

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

__init__(*space:* opytimizer.core.space.Space, *optimizer:* opytimizer.core.optimizer.Optimizer, *function:* opytimizer.core.function.Function, *save_agents: Optional[bool] = 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: *opytimizer.core.space.Space*

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

property optimizer: *opytimizer.core.optimizer.Optimizer*

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

property function: *opytimizer.core.function.Function*

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

property history: *opytimizer.utils.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[*opytimizer.utils.callback.Callback*]*) → None
Wraps the *evaluate* pipeline with its corresponding callbacks.

Parameters **callbacks** – List of callbacks.

update(*callbacks: List[*opytimizer.utils.callback.Callback*]*) → None
Wraps the *update* pipeline with its corresponding callbacks.

Parameters **callback** – List of callbacks.

start(*n_iterations: Optional[int] = 1*, *callbacks: Optional[List[*opytimizer.utils.callback.Callback*]] = 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.

CHAPTER

TWO

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[Union[int, float]]*, *upper_bound: List[Union[int, float]]*, *mapping: Optional[List[str]] = None*)

An Agent class for all optimization techniques.

__init__(*n_variables: int*, *n_dimensions: int*, *lower_bound: List[Union[int, float]]*, *upper_bound: List[Union[int, float]]*, *mapping: Optional[List[str]] = 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: numpy.ndarray

N-dimensional array of positions.

property fit: Union[int, float]

Fitness value.

Type float

property lb: numpy.ndarray

Lower bounds.

property ub: numpy.ndarray

Upper bounds.

property ts: int
Timestamp of the agent.

property mapping: List[str]
Variables mapping.

property mapped_position: Dict[str, numpy.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: numpy.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 opytimizer.core.block

Block.

class opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.core.cell

Cell.

class opytimizer.core.cell.**Cell**(*blocks:* opytimizer.core.block.Block, *edges: Tuple[*opytimizer.core.block.Block, opytimizer.core.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:* opytimizer.core.block.Block, *edges: Tuple[*opytimizer.core.block.Block, opytimizer.core.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 opytimizer.core.function

Single-objective functions.

class opytimizer.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: numpy.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 opytimizer.core.node

Node.

class opytimizer.core.node.**Node**(*name: Union[str, int]*, *category: str*, *value: Optional[numpy.ndarray] = None*, *left: Optional[*opytimizer.core.node.Node*] = None*, *right: Optional[*opytimizer.core.node.Node*] = None*, *parent: Optional[*opytimizer.core.node.Node*] = None*)
A Node instance is used for composing tree-based structures.

__init__(*name: Union[str, int]*, *category: str*, *value: Optional[numpy.ndarray] = None*, *left: Optional[*opytimizer.core.node.Node*] = None*, *right: Optional[*opytimizer.core.node.Node*] = None*, *parent: Optional[*opytimizer.core.node.Node*] = 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: Union[str, int]
Name of the node.

property category: str
Category of the node.

property value: numpy.ndarray
Value of the node.

Type np.array

property left: *opytimizer.core.node.Node*
Pointer to the node's left child.

property right: *opytimizer.core.node.Node*
Pointer to the node's right child.

property parent: *opytimizer.core.node.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: numpy.ndarray
Position after traversing the node.

property post_order: List[*opytimizer.core.node.Node***]**
Traverses the node in post-order.

property pre_order: List[*opytimizer.core.node.Node***]**
Traverses the node in pre-order.

find_node(*position: int*) → *opytimizer.core.node.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:* opytimizer.core.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:* opytimizer.core.node.Node) → numpy.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:* opytimizer.core.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:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 Opytimizer 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 Opytimizer class.

2.7 opytimizer.core.space

Search space.

class opytimizer.core.space.**Space**(*n_agents: Optional[int] = 1*, *n_variables: Optional[int] = 1*, *n_dimensions: Optional[int] = 1*, *lower_bound: Optional[Union[float, List, Tuple, numpy.ndarray]] = 0.0*, *upper_bound: Optional[Union[float, List, Tuple, numpy.ndarray]] = 1.0*, *mapping: Optional[List[str]] = None*)
A Space class for agents, variables and methods related to the search space.

__init__(*n_agents: Optional[int] = 1*, *n_variables: Optional[int] = 1*, *n_dimensions: Optional[int] = 1*, *lower_bound: Optional[Union[float, List, Tuple, numpy.ndarray]] = 0.0*, *upper_bound: Optional[Union[float, List, Tuple, numpy.ndarray]] = 1.0*, *mapping: Optional[List[str]] = 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: numpy.ndarray
Minimum possible values.

property ub: numpy.ndarray
Maximum possible values.

property mapping: List[str]
Variables mapping.

property agents: List[*opytimizer.core.agent.Agent*]
Agents that belongs to the space.

Type list

property best_agent: *opytimizer.core.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 opytimizer modules.

CHAPTER

THREE

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: Optional[float] = 0.0*)

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

__init__(*pointer: List[callable]*, *constraints: List[callable]*, *penalty: Optional[float] = 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: numpy.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 opytimizer.functions.multi_objective

3.2.1 opytimizer.functions.multi_objective.standard

Standard multi-objective functions.

class opytimizer.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: numpy.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 opytimizer.functions.multi_objective.weighted

Multi-objective weighted functions.

class opytimizer.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: numpy.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 opytimizer modules. Functions package for all common opytimizer modules.

CHAPTER

FOUR

OPYTIMIZER.MATH

Just because we are computing stuff does not means 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: Optional[float] = 0.0*, *size: Optional[int] = 1*) → numpy.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: Optional[int] = 1*, *probs: Optional[numpy.ndarray] = None*, *size: Optional[int] = 1*) → numpy.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: Optional[float] = 0.1*, *size: Optional[int] = 1*) → numpy.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 opytimizer.math.general

General-based mathematical functions.

opytimizer.math.general.**euclidean_distance**(*x: numpy.ndarray*, *y: numpy.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)

opytimizer.math.general.**kmeans**(*x: numpy.ndarray*, *n_clusters: Optional[int] = 1*, *max_iterations: Optional[int] = 100*, *tol: Optional[float] = 0.0001*) → numpy.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)

opytimizer.math.general.**n_wise**(*x: List[Any]*, *size: Optional[int] = 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)

opytimizer.math.general.**tournament_selection**(*fitness: List[float]*, *n: int*, *size: Optional[int] = 2*) → numpy.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)

opytimizer.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 opytimizer.math.complex

4.4 opytimizer.math.random

Random-based mathematical generators.

opytimizer.math.random.**generate_binary_random_number**(*size: Optional[int] = 1*) → numpy.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)

opytimizer.math.random.**generate_exponential_random_number**(*scale: Optional[float] = 1.0*, *size: Optional[int] = 1*) → numpy.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: Optional[float] = 1.0*, *scale: Optional[float] = 1.0*, *size: Optional[int] = 1*) → numpy.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: Optional[int] = 0*, *high: Optional[int] = 1*, *exclude_value: Optional[int] = None*, *size: Optional[int] = None*) → numpy.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: Optional[float] = 0.0*, *high: Optional[float] = 1.0*, *size: Optional[int] = 1*) → numpy.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: Optional[float] = 0.0*, *variance: Optional[float] = 1.0*, *size: Optional[int] = 1*) → numpy.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.

CHAPTER

FIVE

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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property S: numpy.ndarray
Somersault foraging.

_cyclone_foraging(*agents: List[opytimizer.core.agent.Agent]*, *best_position: numpy.ndarray*, *i: int*, *iteration: int*, *n_iterations: int*) → numpy.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[*opytimizer.core.agent.Agent*]*, *best_position: numpy.ndarray*, *i: int*) → numpy.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: numpy.ndarray*, *best_position: numpy.ndarray*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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.bpso

Boolean Particle Swarm Optimization.

class opytimizer.optimizers.boolean.bpso.**BPSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property c1: numpy.ndarray
Cognitive constant.

property c2: numpy.ndarray
Social constant.

property local_position: numpy.ndarray
Array of local positions.

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

evaluate(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.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 opytimizer.optimizers.boolean.umda

Univariate Marginal Distribution Algorithm.

class opytimizer.optimizers.boolean.umda.**UMDA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*) → numpy.ndarray
Calculates probabilities based on pre-selected agents' variables occurrence (eq. 47).

Parameters **agents** – List of pre-selected agents.

Returns Probability of variables occurence.

Return type (np.ndarray)

_sample_position(*probs: numpy.ndarray*) → numpy.ndarray
Samples new positions according to their probability of ocurrence (eq. 53).

Parameters **probs** – Array of probabilities.

Returns New sampled position.

Return type (np.ndarray)

update(*space:* opytimizer.core.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 opytimizer modules. It contains implementations of boolean-based optimizers.

5.2 opytimizer.optimizers.evolutionary

5.2.1 opytimizer.optimizers.evolutionary.bsa

Backtracking Search Optimization Algorithm.

class opytimizer.optimizers.evolutionary.bsa.**BSA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]
List of historical agents.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_permute(*agents: List[*opytimizer.core.agent.Agent*]*) → None
Performs the permuting operator.

Parameters **agents** – List of agents.

_mutate(*agents: List[*opytimizer.core.agent.Agent*]*) → List[*opytimizer.core.agent.Agent*]
Performs the mutation operator.

Parameters **agents** – List of agents.

Returns A list holding the trial agents.

Return type (List[*Agent*])

_crossover(*agents: List[*opytimizer.core.agent.Agent*]*, *trial_agents: List[*opytimizer.core.agent.Agent*]*) → None
Performs the crossover operator.

Parameters

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

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.de.**DE**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.agent.Agent, *alpha:* opytimizer.core.agent.Agent, *beta:* opytimizer.core.agent.Agent, *gamma:* opytimizer.core.agent.Agent) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.ep

Evolutionary Programming.

class opytimizer.optimizers.evolutionary.ep.**EP**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray

Array of strategies.

compile(*space:* opytimizer.core.space.Space) → None

Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_mutate_parent(*agent:* opytimizer.core.agent.Agent, *index: int*, *function:* opytimizer.core.function.Function) → *opytimizer.core.agent.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: numpy.ndarray*, *upper_bound: numpy.ndarray*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.es

Evolution Strategies.

class opytimizer.optimizers.evolutionary.es.**ES**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray

Array of strategies.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_mutate_parent(*agent:* opytimizer.core.agent.Agent, *index: int*, *function:* opytimizer.core.function.Function) → *opytimizer.core.agent.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*) → numpy.ndarray
Updates the strategy (eq. 5-10).

Parameters **index** – Index of current agent.

Returns The updated strategy.

Return type (np.ndarray)

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.foa

Forest Optimization Algorithm.

class opytimizer.optimizers.evolutionary.foa.**FOA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_local_seeding(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space) → List[*opytimizer.core.agent.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:* opytimizer.core.space.Space, *function:* opytimizer.core.function.Function, *candidate: List[*opytimizer.core.agent.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.agent.Agent, *mother:* opytimizer.core.agent.Agent) → Tuple[*opytimizer.core.agent.Agent*, *opytimizer.core.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:* opytimizer.core.agent.Agent, *beta:* opytimizer.core.agent.Agent) → Tuple[*opytimizer.core.agent.Agent*, *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.gp

Genetic Programming.

class opytimizer.optimizers.evolutionary.gp.**GP**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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 prunning_ratio: float
Nodes' prunning 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 prunned nodes.

Return type (int)

_reproduction(*space:* opytimizer.spaces.tree.TreeSpace) → None
Reproducts a number of individuals pre-selected through a tournament procedure (p. 99).

Parameters **space** – A TreeSpace object.

_mutation(*space:* opytimizer.spaces.tree.TreeSpace) → None
Mutates a number of individuals pre-selected through a tournament procedure.

Parameters **space** – A TreeSpace object.

_mutate(*space:* opytimizer.spaces.tree.TreeSpace, *tree:* opytimizer.core.node.Node, *max_nodes: int*) → *opytimizer.core.node.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:* opytimizer.spaces.tree.TreeSpace) → None
Crossover a number of individuals pre-selected through a tournament procedure (p. 101).

Parameters **space** – A TreeSpace object.

_cross(*father:* opytimizer.core.node.Node, *mother:* opytimizer.core.node.Node, *max_father: int*, *max_mother: int*) → Tuple[*opytimizer.core.node.Node*, *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space) → None
Wraps Genetic Programming over all trees and variables.

Parameters **space** – TreeSpace containing agents and update-related information.

5.2.8 opytimizer.optimizers.evolutionary.hs

Harmony Search-based algorithms.

class opytimizer.optimizers.evolutionary.hs.HS(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.hs.**IHS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.hs.**GHS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

_generate_new_harmony(*agents: List[*opytimizer.core.agent.Agent*]*) → *opytimizer.core.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 opytimizer.optimizers.evolutionary.hs.**SGHS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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. Neurocomputing (2010).

__init__(*params: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property pm: float
Mutation probability.

_generate_new_harmony(*best:* opytimizer.core.agent.Agent, *worst:* opytimizer.core.agent.Agent) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None

Initialization method.

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

_generate_opposition_harmony(*new_agent:* opytimizer.core.agent.Agent, *agents:* *List[*opytimizer.core.agent.Agent*]*) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function) → *opytimizer.core.agent.Agent*
Reproduces and flowers a seed into a new offpsring.

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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.evolutionary.rra

Runner-Root Algorithm.

class opytimizer.optimizers.evolutionary.rra.**RRA**(*params: Optional[Dict[str, Any]] = None*)
An RRA class, inherited from Optimizer.

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

References

F. Merrikh-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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.function.Function, *is_large: Optional[bool] = 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: Optional[float] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 optimizers.

5.3 opytimizer.optimizers.misc

5.3.1 opytimizer.optimizers.misc.aoa

Arithmetic Optimization Algorithm.

class opytimizer.optimizers.misc.aoa.**AOA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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 opytimizer.optimizers.misc.cem

Cross-Entropy Method.

class opytimizer.optimizers.misc.cem.**CEM**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of means.

property std: numpy.ndarray
Array of standard deviations.

compile(*space:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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: numpy.ndarray*) → numpy.ndarray
Calculates and updates mean.

Parameters **updates** – An array of updates' positions.

Returns The new mean values.

Return type (np.ndarray)

_update_std(*updates: numpy.ndarray*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.misc.doa

Darcy Optimization Algorithm.

class opytimizer.optimizers.misc.doa.**DOA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property r: float
Chaos multiplier.

property chaotic_map: numpy.ndarray
Array of chaotic maps.

compile(*space:* opytimizer.core.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:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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 opytimizer.optimizers.misc.nds

Non-Dominated Sorting.

class opytimizer.optimizers.misc.nds.**NDS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of domination counts.

property set: numpy.ndarray
Array of dominating set.

property status: numpy.ndarray
Array of pareto status.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_compare_domination(*agent_i:* opytimizer.core.agent.Agent, *agent_j:* opytimizer.core.agent.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:* opytimizer.core.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 opytimizer modules. It contains implementations of miscellaneous-based optimizers.

5.4 opytimizer.optimizers.population

5.4.1 opytimizer.optimizers.population.aeo

Artificial Ecosystem-based Optimization.

class opytimizer.optimizers.population.aeo.**AEO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

_production(*agent:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *iteration: int*, *n_iterations: int*) → *opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *producer:* opytimizer.core.agent.Agent, *C: float*) → *opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *producer:* opytimizer.core.agent.Agent, *consumer:* opytimizer.core.agent.Agent, *C: float*) → *opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *consumer:* opytimizer.core.agent.Agent, *C:* *float*) → *opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.ao

Aquila Optimizer.

class opytimizer.optimizers.population.ao.**AO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *index: int*) → List[*opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*) → None
Transits coyotes between packs (eq. 4).

Parameters **agents** – List of agents.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.epo

Emperor Penguin Optimizer.

class opytimizer.optimizers.population.epo.**EPO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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 opytimizer.optimizers.population.gco

Germinal Center Optimization.

class opytimizer.optimizers.population.gco.**GCO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of lives.

property counter: numpy.ndarray
Array of counters.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_mutate_cell(*agent:* opytimizer.core.agent.Agent, *alpha:* opytimizer.core.agent.Agent, *beta:* opytimizer.core.agent.Agent, *gamma:* opytimizer.core.agent.Agent) → *opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*) → None

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

Parameters **agents** – List of agents.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.gwo

Grey Wolf Optimizer.

class opytimizer.optimizers.population.gwo.**GWO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.hho

Harris Hawks Optimization.

class opytimizer.optimizers.population.hho.**HHO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*, *current_agent:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent) → numpy.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[*opytimizer.core.agent.Agent*]*, *current_agent:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.loa

Lion Optimization Algorithm.

class opytimizer.optimizers.population.loa.**Lion**(*n_variables: int*, *n_dimensions: int*, *lower_bound: Union[List, Tuple, numpy.ndarray]*, *upper_bound: Union[List, Tuple, numpy.ndarray]*, *position: numpy.ndarray*, *fit: float*)

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

__init__(*n_variables: int*, *n_dimensions: int*, *lower_bound: Union[List, Tuple, numpy.ndarray]*, *upper_bound: Union[List, Tuple, numpy.ndarray]*, *position: numpy.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: numpy.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 opytimizer.optimizers.population.loa.**LOA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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[*opytimizer.optimizers.population.loa.Lion*]*) → List[*opytimizer.optimizers.population.loa.Lion*]
Gets all nomad lions.

Parameters **agents** – Agents.

Returns A list of nomad lions.

Return type (List[*Lion*])

_get_pride_lions(*agents: List[*opytimizer.optimizers.population.loa.Lion*]*) → List[List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.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[*opytimizer.optimizers.population.loa.Lion*]*) → None
Move prides to safe locations (s. 2.2.3).

Parameters **prides** – List of prides holding their corresponding lions.

_roaming(*prides: List[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.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[*opytimizer.optimizers.population.loa.Lion*]*, *males: List[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.function.Function) → Tuple[*opytimizer.optimizers.population.loa.Lion*, *opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.function.Function) → *opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *prides: List[List[*opytimizer.optimizers.population.loa.Lion*]]*, *cubs: List[*opytimizer.optimizers.population.loa.Lion*]*) → Tuple[List[*opytimizer.optimizers.population.loa.Lion*], List[List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.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[*opytimizer.optimizers.population.loa.Lion*]*, *function:* opytimizer.core.function.Function) → List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *prides: List[List[*opytimizer.optimizers.population.loa.Lion*]]*) → Tuple[List[*opytimizer.optimizers.population.loa.Lion*], List[List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *prides: List[List[*opytimizer.optimizers.population.loa.Lion*]]*) → Tuple[List[*opytimizer.optimizers.population.loa.Lion*], List[List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]*, *prides: List[List[*opytimizer.optimizers.population.loa.Lion*]]*, *n_agents: List[*opytimizer.core.agent.Agent*]*) → Tuple[List[*opytimizer.optimizers.population.loa.Lion*], List[List[*opytimizer.optimizers.population.loa.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[*opytimizer.optimizers.population.loa.Lion*]]*) → None

Checks if there is at least one male per pride.

Parameters **prides** – List of prides holding their corresponding lions.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.osa

Owl Search Algorithm.

class opytimizer.optimizers.population.osa.**OSA**(*params: Optional[Dict[str, Any]] = 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 novelnature-inspired heuristic paradigm for global optimization. Journal of Intelligent & Fuzzy Systems (2018).

__init__(*params: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property beta: float
Exploration intensity.

update(*space:* opytimizer.core.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 opytimizer.optimizers.population.ppa

Parasitism-Predation Algorithm.

class opytimizer.optimizers.population.ppa.**PPA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.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:* opytimizer.core.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:* opytimizer.core.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:* opytimizer.core.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:* opytimizer.core.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 opytimizer.optimizers.population.pvs

Passing Vehicle Search.

class opytimizer.optimizers.population.pvs.**PVS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.population.rfo

Red Fox Optimization.

class opytimizer.optimizers.population.rfo.**RFO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_rellocation(*agent:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function) → None
Performs the fox rellocation procedure.

Parameters

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

_noticing(*agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer modules. It contains implementations of population-based optimizers.

5.5 opytimizer.optimizers.science

5.5.1 opytimizer.optimizers.science.aig

Algorithm of the Innovative Gunner.

class opytimizer.optimizers.science.aig.**AIG**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.aso

Atom Search Optimization.

class opytimizer.optimizers.science.aso.**ASO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_calculate_mass(*agents: List[*opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *K_agent:* opytimizer.core.agent.Agent, *average: numpy.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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *mass: numpy.ndarray*, *iteration: int*, *n_iterations: int*) → numpy.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:* opytimizer.core.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 opytimizer.optimizers.science.bh

Black Hole.

class opytimizer.optimizers.science.bh.**BH**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None

Initialization method.

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

_update_position(*agents: List[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.eo

Equilibrium Optimizer.

class opytimizer.optimizers.science.eo.**EO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]
Concentrations (agents).

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_calculate_equilibrium(*agents: List[*opytimizer.core.agent.Agent*]*) → None
Calculates the equilibrium concentrations.

Parameters **agents** – List of agents.

_average_concentration(*function:* opytimizer.core.function.Function) → *opytimizer.core.agent.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.esa

Electro-Search Algorithm.

class opytimizer.optimizers.science.esa.**ESA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Orbital radius.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.gsa

Gravitational Search Algorithm.

class opytimizer.optimizers.science.gsa.**GSA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property G: float
Initial gravity.

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_calculate_mass(*agents: List[*opytimizer.core.agent.Agent*]*) → float
Calculates agents' mass (eq. 16).

Parameters **agents** – List of agents.

Returns The agents' mass.

Return type (float)

_calculate_force(*agents: List[*opytimizer.core.agent.Agent*]*, *mass: numpy.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:* opytimizer.core.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 opytimizer.optimizers.science.hgso

Henry Gas Solubility Optimization.

class opytimizer.optimizers.science.hgso.**HGSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of coefficients.

property pressure: numpy.ndarray
Array of pressures.

property constant: numpy.ndarray
Array of constants.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_update_position(*agent:* opytimizer.core.agent.Agent, *cluster_agent:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *solubility: float*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.lsa

Lightning Search Algorithm.

class opytimizer.optimizers.science.lsa.**LSA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray

Array of directions.

compile(*space:* opytimizer.core.space.Space) → None

Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_update_direction(*agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None

Initialization method.

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

property alpha: float

Particle moviment first constant.

property rho: float

Particle moviment second constant.

compile(*space:* opytimizer.core.space.Space) → None

Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]
Thermal memory.

property environment: List[*opytimizer.core.agent.Agent*]
Environmental population.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of flows.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_flow_intensity(*agents: List[*opytimizer.core.agent.Agent*]*) → None
Calculates the intensity of each possible flow (eq. 6).

Parameters **agents** – List of agents.

_raining_process(*agents: List[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent) → None
Performs the raining process (eq. 11-12).

Parameters

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

_update_stream(*agents: List[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.wdo

Wind Driven Optimization.

class opytimizer.optimizers.science.wdo.**WDO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.weo

Water Evaporation Optimization.

class opytimizer.optimizers.science.weo.**WEO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.science.wwo

Water Wave Optimization.

class opytimizer.optimizers.science.wwo.**WWO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of heights.

property length: numpy.ndarray
Array of lengths.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_propagate_wave(*agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function, *index: int*) → *opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function, *j: int*) → *opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*) → None

Updates the wave length of current population.

Parameters **agents** – List of agents.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer modules. It contains implementations of science-based optimizers.

5.6 opytimizer.optimizers.social

5.6.1 opytimizer.optimizers.social.bso

Brain Storm Optimization.

class opytimizer.optimizers.social.bso.**BSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*) → Tuple[numpy.ndarray, numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of lower bounds.

property upper: numpy.ndarray
Array of upper bounds.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.social.isa

Interactive Search Algorithm.

class opytimizer.optimizers.social.isa.**ISA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of velocities.

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

evaluate(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.social.mvpa

Most Valuable Player Algorithm.

class opytimizer.optimizers.social.mvpa.**MVPA**(*params: Optional[Dict[str, Any]] = None*)
A MVPA class, inherited from Optimizer.

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

References

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

__init__(*params: Optional[Dict[str, Any]] = 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:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *index: int*) → List[*opytimizer.core.agent.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of local positions.

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_mean_global_solution(*alpha: numpy.ndarray*, *beta: numpy.ndarray*, *gamma: numpy.ndarray*) → numpy.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: numpy.ndarray*, *index: int*) → numpy.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: numpy.ndarray*, *mean: numpy.ndarray*, *index: int*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer modules. It contains implementations of human social behavior-based optimizers.

5.7 opytimizer.optimizers.swarm

5.7.1 opytimizer.optimizers.swarm.abc

Artificial Bee Colony.

class opytimizer.optimizers.swarm.abc.**ABC**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property n_trials: int
Number of trial limits.

property trial: numpy.ndarray
Array of trial.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_evaluate_location(*agent:* opytimizer.core.agent.Agent, *neighbour:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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 neightbour agent.
- **function** – A function object.
- **index** – Index of trial.

_send_employee(*agents: List[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.function.Function) → None
Sends scout bees to scout for new possible food sources.

Parameters

- **agents** – List of agents.
- **function** – A function object.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.agent.Agent, *neighbour:* opytimizer.core.agent.Agent, *function:* opytimizer.core.function.Function) → Tuple[*opytimizer.core.agent.Agent*, bool]
Flies to a new location according to the flight mode (eq. 1).

Parameters

- **agent** – Current agent.
- **neighbour** – Selected neigbour.
- **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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.af

Artificial Flora.

class opytimizer.optimizers.swarm.af.**AF**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.ba

Bat Algorithm.

class opytimizer.optimizers.swarm.ba.**BA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of frequencies.

property velocity: numpy.ndarray
Array of velocities.

property loudness: numpy.ndarray
Array of loudnesses.

property pulse_rate: numpy.ndarray
Array of pulse rates.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of fragrances.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_best_movement(*agent_position: numpy.ndarray*, *best_position: numpy.ndarray*, *fragrance: numpy.ndarray*, *random: float*) → numpy.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: numpy.ndarray*, *j_position: numpy.ndarray*, *k_position: numpy.ndarray*, *fragrance: numpy.ndarray*, *random: float*) → numpy.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:* opytimizer.core.space.Space) → None
Wraps Butterfly Optimization Algorithm over all agents and variables.

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

5.7.6 opytimizer.optimizers.swarm.bwo

Black Widow Optimization.

class opytimizer.optimizers.swarm.bwo.**BWO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.agent.Agent, *x2:* opytimizer.core.agent.Agent) → Tuple[*opytimizer.core.agent.Agent*, *opytimizer.core.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:* opytimizer.core.agent.Agent) → *opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent) → List[*opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *prob: float*) → List[*opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *new_agents: List[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.csa

Crow Search Algorithm.

class opytimizer.optimizers.swarm.csa.**CSA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of memories.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

evaluate(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space) → None
Wraps Crow Search Algorithm over all agents and variables.

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

5.7.9 opytimizer.optimizers.swarm.eho

Elephant Herding Optimization.

class opytimizer.optimizers.swarm.eho.**EHO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*, *index: int*) → List[*opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *centers: numpy.ndarray*, *function:* opytimizer.core.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[*opytimizer.core.agent.Agent*]*) → None
Performs the separating operator.

Parameters **agents** – List of agents.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property x_axis: List[*opytimizer.core.agent.Agent*]
x axis.

property y_axis: List[*opytimizer.core.agent.Agent*]
y axis.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray*, *best_position: numpy.ndarray*) → numpy.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: numpy.ndarray*, *k_position: numpy.ndarray*, *l_position: numpy.ndarray*, *epsilon: float*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.fso

Flying Squirrel Optimizer.

class opytimizer.optimizers.swarm.fso.**FSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None

Initialization method.

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

property beta: float

Lévy distribution parameter.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.goa

Grasshopper Optimization Algorithm.

class opytimizer.optimizers.swarm.goa.**GOA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.js

Jellyfish Search-based algorithms.

class opytimizer.optimizers.swarm.js.**JS**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property eta: float
Chaotic map coefficient.

property beta: float
Distribution coeffiecient.

property gamma: float
Motion coeffiecient.

_initialize_chaotic_map(*agents: List[*opytimizer.core.agent.Agent*]*) → None
Initializes a set of agents using a logistic chaotic map.

Parameters **agents** – List of agents.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_ocean_current(*agents: List[*opytimizer.core.agent.Agent*]*, *best_agent:* opytimizer.core.agent.Agent) → numpy.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: numpy.ndarray*, *ub: numpy.ndarray*) → numpy.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: opytimizer.core.agent.Agent*, *agent_j: opytimizer.core.agent.Agent*) → numpy.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: opytimizer.core.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 opytimizer.optimizers.swarm.js.NBJS(*params: Optional[Dict[str, Any]] = None*)
An NBJS class, inherited from JS.

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

References

Publication pending.

__init__(*params: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

_motion_a(*lb: numpy.ndarray*, *ub: numpy.ndarray*) → numpy.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 opytimizer.optimizers.swarm.kh

Krill Herd.

class opytimizer.optimizers.swarm.kh.KH(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of motions.

property foraging: numpy.ndarray
Array of foragings.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_food_location(*agents: List[*opytimizer.core.agent.Agent*]*, *function:* opytimizer.core.function.Function) → *opytimizer.core.agent.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[*opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *idx: int*, *sensing_distance: float*, *eucl_distance: List[float]*) → List[*opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *worst:* opytimizer.core.agent.Agent, *best:* opytimizer.core.agent.Agent, *neighbours: List[*opytimizer.core.agent.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:* opytimizer.core.agent.Agent, *worst:* opytimizer.core.agent.Agent, *best:* opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *idx: int*, *iteration: int*, *n_iterations: int*, *motion: numpy.ndarray*) → numpy.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:* opytimizer.core.agent.Agent, *worst:* opytimizer.core.agent.Agent, *best:* opytimizer.core.agent.Agent, *food: numpy.ndarray*, *C_food: float*) → numpy.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:* opytimizer.core.agent.Agent, *worst:* opytimizer.core.agent.Agent, *best:* opytimizer.core.agent.Agent) → numpy.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[*opytimizer.core.agent.Agent*]*, *idx: int*, *iteration: int*, *n_iterations: int*, *food: numpy.ndarray*, *foraging: numpy.ndarray*) → numpy.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[*opytimizer.core.agent.Agent*]*, *idx: int*, *iteration: int*, *n_iterations: int*, *food: numpy.ndarray*, *motion: numpy.ndarray*, *foraging: numpy.ndarray*) → numpy.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[*opytimizer.core.agent.Agent*]*, *idx: int*) → *opytimizer.core.agent.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[*opytimizer.core.agent.Agent*]*, *idx: int*) → *opytimizer.core.agent.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.mfo

Moth-Flame Optimization.

class opytimizer.optimizers.swarm.mfo.**MFO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property b: float
Spiral constant.

update(*space:* opytimizer.core.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 opytimizer.optimizers.swarm.mrfo

Manta Ray Foraging Optimization.

class opytimizer.optimizers.swarm.mrfo.**MRFO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property S: float
Somersault foraging.

_cyclone_foraging(*agents: List[*opytimizer.core.agent.Agent*]*, *best_position: numpy.ndarray*, *i: int*, *iteration: int*, *n_iterations: int*) → numpy.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[*opytimizer.core.agent.Agent*]*, *best_position: numpy.ndarray*, *i: int*) → numpy.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: numpy.ndarray*, *best_position: numpy.ndarray*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.pio

Pigeon-Inspired Optimization.

class opytimizer.optimizers.swarm.pio.**PIO**(*params: Optional[Dict[str, Any]] = 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 optimizerfor air robot path planning. International Journal of IntelligentComputing and Cybernetics (2014).

__init__(*params: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of pulse rates.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_calculate_center(*agents: List[*opytimizer.core.agent.Agent*]*) → numpy.ndarray
Calculates the center position (eq. 8).

Parameters **agents** – List of agents.

Returns The center position.

Return type (np.ndarray)

_update_center_position(*position: numpy.ndarray*, *center: numpy.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:* opytimizer.core.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 opytimizer.optimizers.swarm.pso

Particle Swarm Optimization-based algorithms.

class opytimizer.optimizers.swarm.pso.**PSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of velocities.

property velocity: numpy.ndarray
Array of velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

evaluate(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.space.Space) → None
Wraps Particle Swarm Optimization over all agents and variables.

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

class opytimizer.optimizers.swarm.pso.**AIWPSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]*) → None
Computes the particles' success for updating inertia weight (eq. 16).

Parameters **agents** – List of agents.

update(*space:* opytimizer.core.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 opytimizer.optimizers.swarm.pso.**RPSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property mass: numpy.ndarray
Array of masses.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space) → None
Wraps Relativistic Particle Swarm Optimization over all agents and variables.

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

class opytimizer.optimizers.swarm.pso.**SAVPSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

update(*space:* opytimizer.core.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 opytimizer.optimizers.swarm.pso.**VPSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property v_velocity: numpy.ndarray
Array of vertical velocities.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.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 opytimizer.optimizers.swarm.sbo

Satin Bowerbird Optimizer.

class opytimizer.optimizers.swarm.sbo.**SBO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

update(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.sca

Sine Cosine Algorithm.

class opytimizer.optimizers.swarm.sca.**SCA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray*, *best_position: numpy.ndarray*, *r1: float*, *r2: float*, *r3: float*, *r4: float*) → numpy.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: opytimizer.core.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 opytimizer.optimizers.swarm.sfo

Sailfish Optimizer.

class opytimizer.optimizers.swarm.sfo.**SFO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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[*opytimizer.core.agent.Agent*]
List of sardines.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

_generate_random_agent(*agent:* opytimizer.core.agent.Agent) → *opytimizer.core.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:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *best_sardine:* opytimizer.core.agent.Agent, *lambda_i: float*) → numpy.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

_mutualism(*agent_i:* opytimizer.core.agent.Agent, *agent_j:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.agent.Agent, *agent_j:* opytimizer.core.agent.Agent, *best_agent:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.agent.Agent, *agent_j:* opytimizer.core.agent.Agent, *function:* opytimizer.core.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:* opytimizer.core.space.Space, *function:* opytimizer.core.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 opytimizer.optimizers.swarm.ssa

Salp Swarm Algorithm.

class opytimizer.optimizers.swarm.ssa.**SSA**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

update(*space:* opytimizer.core.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 opytimizer.optimizers.swarm.sso

Simplified Swarm Optimization.

class opytimizer.optimizers.swarm.sso.**SSO**(*params: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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: numpy.ndarray
Array of local positions.

compile(*space:* opytimizer.core.space.Space) → None
Compiles additional information that is used by this optimizer.

Parameters **space** – A Space object containing meta-information.

evaluate(*space:* opytimizer.core.space.Space, *function:* opytimizer.core.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:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = 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:* opytimizer.core.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: Optional[Dict[str, Any]] = 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: Optional[Dict[str, Any]] = None*) → None
Initialization method.

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

property b: float
Logarithmic spiral.

_generate_random_agent(*agent:* opytimizer.core.agent.Agent) → *opytimizer.core.agent.Agent*
Generates a new random-based agent.

Parameters **agent** – Agent to be copied.

Returns Random-based agent.

Return type (*Agent*)

update(*space:* opytimizer.core.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 opytimizer modules. It contains implementations of swarm-based optimizers.
An optimizers package for all common opytimizer modules. It contains specific packages of every optimization taxonomy covered by opytimizer.

CHAPTER

SIX

OPYTIMIZER.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 opytimizer.spaces.boolean

Boolean-based search space.

class opytimizer.spaces.boolean.**BooleanSpace**(*n_agents: int*, *n_variables: int*, *mapping: Optional[List[str]] = None*)

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

__init__(*n_agents: int*, *n_variables: int*, *mapping: Optional[List[str]] = 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 opytimizer.spaces.graph

Graph-based search space.

6.3 opytimizer.spaces.grid

Grid-based search space.

class opytimizer.spaces.grid.**GridSpace**(*n_variables: int*, *step: Union[float, List, Tuple, numpy.ndarray]*, *lower_bound: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *mapping: Optional[List[str]] = None*)

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

__init__(*n_variables: int*, *step: Union[float, List, Tuple, numpy.ndarray]*, *lower_bound: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *mapping: Optional[List[str]] = 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: numpy.ndarray
Step size of each variable.

property grid: numpy.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 opytimizer.spaces.hyper_complex

Hypercomplex-based search space.

class opytimizer.spaces.hyper_complex.HyperComplexSpace(*n_agents: int*, *n_variables: int*, *n_dimensions: int*, *mapping: Optional[List[str]] = 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: Optional[List[str]] = 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 opytimizer.spaces.pareto

Pareto-based search space.

class opytimizer.spaces.pareto.**ParetoSpace**(*data_points: numpy.ndarray*, *mapping: Optional[List[str]] = None*)

A ParetoSpace class for agents, variables and methods related to the pareto-frontier search space.

__init__(*data_points: numpy.ndarray*, *mapping: Optional[List[str]] = None*) → None

Initialization method.

Parameters

- **data_points** – Pre-defined data points.
- **mapping** – String-based identifiers for mapping variables' names.

_load_agents(*data_points: numpy.ndarray*) → None

Loads agents from pre-defined data points.

Parameters **data_points** – Pre-defined data points.

build(*data_points: numpy.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: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *mapping: Optional[List[str]] = None*)

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

__init__(*n_agents: int*, *n_variables: int*, *lower_bound: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *mapping: Optional[List[str]] = 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: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *n_terminals: Optional[int] = 1*, *min_depth: Optional[int] = 1*, *max_depth: Optional[int] = 3*, *functions: Optional[List[str]] = None*, *mapping: Optional[List[str]] = 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: Union[float, List, Tuple, numpy.ndarray]*, *upper_bound: Union[float, List, Tuple, numpy.ndarray]*, *n_terminals: Optional[int] = 1*, *min_depth: Optional[int] = 1*, *max_depth: Optional[int] = 3*, *functions: Optional[List[str]] = None*, *mapping: Optional[List[str]] = 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[*opytimizer.core.node.Node*]

Trees (derived from the Node class).

property best_tree: *opytimizer.core.node.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: Optional[int] = 1*, *max_depth: Optional[int] = 3*) → *opytimizer.core.node.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.

CHAPTER

SEVEN

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.utils.callback.Opytimizer*) → None

Performs a callback whenever a task begins.

Parameters **opt_model** – An instance of the optimization model.

on_task_end(*opt_model: opytimizer.utils.callback.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.utils.callback.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.utils.callback.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 opytimizer.utils.callback.**CallbackVessel**(*callbacks: List[*opytimizer.utils.callback.Callback*]*)
Wraps multiple callbacks in an ready-to-use class.

__init__(*callbacks: List[*opytimizer.utils.callback.Callback*]*) → None
Initialization method.

Parameters **callbacks** – List of Callback-based childs.

property callbacks: List[*opytimizer.utils.callback.Callback*]
List of Callback-based childs.

on_task_begin(*opt_model: opytimizer.utils.callback.Opytimizer*) → None
Performs a list of callbacks whenever a task begins.

Parameters **opt_model** – An instance of the optimization model.

on_task_end(*opt_model: opytimizer.utils.callback.Opytimizer*) → 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: opytimizer.utils.callback.Opytimizer*) → 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: opytimizer.utils.callback.Opytimizer*) → 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 opytimizer.utils.callback.**CheckpointCallback**(*file_path: Optional[str] = None*, *frequency: Optional[int] = 0*)
A CheckpointCallback class that handles additional logging and model's checkpointing.

__init__(*file_path: Optional[str] = None*, *frequency: Optional[int] = 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: opytimizer.utils.callback.Opytimizer*) → None
Performs a callback whenever an iteration ends.

Parameters

- **iteration** – Current iteration.
- **opt_model** – An instance of the optimization model.

class opytimizer.utils.callback.**DiscreteSearchCallback**(*allowed_values: Optional[List[Union[int, float]]] = None*)
A DiscreteSearchCallback class that handles mapping floating-point variables to discrete values.

__init__(*allowed_values: Optional[List[Union[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[Union[int, float]]
Allowed values between lower and upper bounds.

on_task_begin(*opt_model: opytimizer.utils.callback.Opytimizer*) → 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 opytimizer.utils.constant

Constants.

7.3 opytimizer.utils.exception

Exceptions.

exception opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.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 opytimizer.utils.history

History-based object that helps in saving the optimization history.

class opytimizer.utils.history.**History**(*save_agents: Optional[bool] = 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: Optional[bool] = 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*) → Union[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: Optional[Tuple[int, ...]] = 0*) → numpy.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 opytimizer.utils.logging

Logging-based methods and helpers.

class opytimizer.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**() → logging.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**() → logging.handlers.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*) → *opytimizer.utils.logging.Logger*
Gets a logger and make it avaliable for further use.

Parameters **logger_name** – The name of the logger.

Returns Logger instance.

Return type (*Logger*)

Utility package for all common opytimizer modules.

CHAPTER

EIGHT

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: Optional[List[str]] = None*, *title: Optional[str] = ''*, *subtitle: Optional[str] = ''*, *xlabel: Optional[str] = 'iteration'*, *ylabel: Optional[str] = 'value'*, *grid: Optional[bool] = True*, *legend: Optional[bool] = 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: numpy.ndarray*, *title: Optional[str] = ''*, *subtitle: Optional[str] = ''*, *style: Optional[str] = 'winter'*, *colorbar: Optional[bool] = 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 opytimizer modules.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

Symbols

A

B

C

D

E

F

G

N

O

P

Q

R

S

T

U

V

W

X

Y

Z