ENOPPY

Release 0.1.1

Thieu

Dec 18, 2023

INTRODUCTION:

ENOPPY (ENgineering Optimization Problems in PYthon) is the largest python library for real-world engineering optimization problems. Contains all real-world engineering problems from CEC competitions and research papers.

- **Free software:** GNU General Public License (GPL) V3 license
- **Total problems**: > 50 problems
- **Documentation:** https://enoppy.readthedocs.io/en/latest/
- **Python versions:** 3.7.x, 3.8.x, 3.9.x, 3.10.x, 3.11.x
- **Dependencies:** numpy, scipy

CHAPTER

ONE

FEATURES

- Our library provides all state-of-the-art engineering optimization problems.
- We have implemented all problems using Numpy to increase the speed of the algorithms.

1.1 Quick Start

1.1.1 Installation

- Install the current PyPI release:

```
$ pip install enoppy==0.1.1
```

- Install directly from source code:

```
$ git clone https://github.com/thieu1995/enoppy.git
$ cd enoppy
$ python setup.py install
```

1.1.2 Lib's structure

Current's structure:

```
docs
examples
enoppy
   paper_based
      pdo_2022.py
      rwco_2020.py
   problem_based
      chemical.py
      mechanism.py
   utils
      validator.py
      visualize.py
   __init__.py
   engineer.py
README.md
setup.py
```

1.1.3 Usage

After installation, you can import ENOPPY as any other Python module:

```
$ python
>>> import enoppy
>>> enoppy.__version__
```

Let's go through some examples.

1.1.4 Examples

How to get the problem and use it:

```python
from enoppy.paper_based.moeosma_2023 import SpeedReducerProblem
# SRP = SpeedReducerProblem
# SP = SpringProblem
# HTBP = HydrostaticThrustBearingProblem
# VPP = VibratingPlatformProblem
# CSP = CarSideImpactProblem
# WRMP = WaterResourceManagementProblem
# BCP = BulkCarriersProblem
# MPBPP = MultiProductBatchPlantProblem

srp_prob = SpeedReducerProblem()
print("Lower bound for this problem: ", srp_prob.lb)
print("Upper bound for this problem: ", srp_prob.ub)
x0 = srp_prob.create_solution()
print("Get the objective values of x0: ", srp_prob.get_objs(x0))
print("Get the constraint values of x0: ", srp_prob.get_cons(x0))
print("Evaluate with default penalty function: ", srp_prob.evaluate(x0))
```

Design my own penalty function:

```python
import numpy as np
from enoppy.paper_based.moeosma_2023 import HTBP
# HTBP = HydrostaticThrustBearingProblem

def penalty_func(list_objectives, list_constraints):
   list_constraints[list_constraints < 0] = 0
   return np.sum(list_objectives) + 1e5 * np.sum(list_constraints**2)

htbp_prob = HTBP(f_penalty=penalty_func)
print("Lower bound for this problem: ", htbp_prob.lb)
print("Upper bound for this problem: ", htbp_prob.ub)
x0 = htbp_prob.create_solution()
print("Get the objective values of x0: ", htbp_prob.get_objs(x0))
print("Get the constraint values of x0: ", htbp_prob.get_cons(x0))
print("Evaluate with default penalty function: ", htbp_prob.evaluate(x0))
```

For more usage examples please look at [examples](/examples) folder.

1.2 ENOPPY Library

1.2.1 enoppy.paper_based

1.2.1.1 enoppy.paper_based.ihaoavoa_2022

enoppy.paper_based.ihaoavoa_2022.**CBP**

alias of *enoppy.paper_based.ihaoavoa_2022.CantileverBeamProblem*

class enoppy.paper_based.ihaoavoa_2022.**CantileverBeamProblem**(*f_penalty=None*)

Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4, x5]

evaluate(*x*)

Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Cantilever beam design problem'

enoppy.paper_based.ihaoavoa_2022.**REBP**

alias of *enoppy.paper_based.ihaoavoa_2022.RollingElementBearingProblem*

class enoppy.paper_based.ihaoavoa_2022.**RollingElementBearingProblem**(*f_penalty=None*)

Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10] = [Dm, Db, Z, fi, f0, Kdmin, Kdmax, theta, e, C]

amend_position(*x, lb=None, ub=None*)

Amend position to fit the format of the problem

Parameters x (*np.ndarray*) – The current position (solution)

evaluate(*x*)

Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Rolling element bearing design problem'

enoppy.paper_based.ihaoavoa_2022.**SRP**
alias of *enoppy.paper_based.ihaoavoa_2022.SpeedReducerProblem*

class enoppy.paper_based.ihaoavoa_2022.**SpeedReducerProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4, x5, x6, x7]

Ref: https://www.hindawi.com/journals/mpe/2013/419043/

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Speed reducer design problem'

enoppy.paper_based.ihaoavoa_2022.**TCSP**
alias of *enoppy.paper_based.ihaoavoa_2022.TensionCompressionSpringProblem*

class enoppy.paper_based.ihaoavoa_2022.**TensionCompressionSpringProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3] = [d, D, N]

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Tension/compression spring design problem'

enoppy.paper_based.ihaoavoa_2022.**WBP**
alias of *enoppy.paper_based.ihaoavoa_2022.WeldedBeamProblem*

class enoppy.paper_based.ihaoavoa_2022.**WeldedBeamProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4] = [h, l, t, b]

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Welded beam design problem'

1.2.1.2 enoppy.paper_based.moeosma_2023

enoppy.paper_based.moeosma_2023.**BCP**
alias of *enoppy.paper_based.moeosma_2023.BulkCarriersProblem*

class enoppy.paper_based.moeosma_2023.**BulkCarriersProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [L, B, D, T, Vk, CB] = [x1, x2, x3, x4, x5, x6]

Original Ref:

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Bulk carriers design problem'

enoppy.paper_based.moeosma_2023.**CSP**
alias of *enoppy.paper_based.moeosma_2023.CarSideImpactProblem*

class enoppy.paper_based.moeosma_2023.**CarSideImpactProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4, x5, x6, x7]

Original Ref:

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Car side impact design problem'

enoppy.paper_based.moeosma_2023.**HTBP**
alias of *enoppy.paper_based.moeosma_2023.HydrostaticThrustBearingProblem*

class enoppy.paper_based.moeosma_2023.**HydrostaticThrustBearingProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4] = [R, R0, mu, Q]

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Hydrostatic thrust bearing design problem'

enoppy.paper_based.moeosma_2023.**MPBPP**
alias of *enoppy.paper_based.moeosma_2023.MultiProductBatchPlantProblem*

class enoppy.paper_based.moeosma_2023.**MultiProductBatchPlantProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [N1, N2, N3, V1, V2, V3, TL1, TL2, B1, B2] = [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10]

Original Ref:

amend_position(*x, lb=None, ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Multi-product batch plant problem'

enoppy.paper_based.moeosma_2023.**SP**
alias of *enoppy.paper_based.moeosma_2023.SpringProblem*

enoppy.paper_based.moeosma_2023.**SRP**
alias of *enoppy.paper_based.moeosma_2023.SpeedReducerProblem*

class enoppy.paper_based.moeosma_2023.**SpeedReducerProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4, x5, x6, x7] = [b, m, z, l1, l2, d1, d2]

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Speed Reducer Design Problem'

class enoppy.paper_based.moeosma_2023.**SpringProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3] = [d, D, N]

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Spring Design Problem'

enoppy.paper_based.moeosma_2023.**VPP**
alias of *enoppy.paper_based.moeosma_2023.VibratingPlatformProblem*

`class enoppy.paper_based.moeosma_2023.VibratingPlatformProblem(f_penalty=None)`

Bases: `enoppy.engineer.Engineer`

x = [d1, d2, d3, b, L] = [x0, x1, x2, x3, x4]

Original Ref: On improving multiobjective genetic algorithms for design optimization

evaluate(*x*)

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Vibrating platform design problem'

`enoppy.paper_based.moeosma_2023.WRMP`

alias of `enoppy.paper_based.moeosma_2023.WaterResourceManagementProblem`

`class enoppy.paper_based.moeosma_2023.WaterResourceManagementProblem(f_penalty=None)`

Bases: `enoppy.engineer.Engineer`

x = [x1, x2, x3]

Original Ref:

evaluate(*x*)

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Water resource management problem'

1.2.1.3 enoppy.paper_based.pdo_2022

`enoppy.paper_based.pdo_2022.CBD`

alias of `enoppy.paper_based.pdo_2022.CantileverBeamProblem`

`enoppy.paper_based.pdo_2022.CBHD`

alias of `enoppy.paper_based.pdo_2022.CorrugatedBulkheadProblem`

`enoppy.paper_based.pdo_2022.CSP`

alias of `enoppy.paper_based.pdo_2022.CompressionSpringProblem` `class enoppy.paper_based.pdo_2022.CantileverBeamProblem(f_penalty=None)`

Bases: `enoppy.engineer.Engineer`

Minimize a cantilever beam's weight. [x1, x2, x3, x4, x5]

`evaluate(x)`

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`

Compute the values of the constraint functions for a given set of input values.

`get_objs(x)`

Compute the values of the objective functions for a given set of input values.

`name = 'Cantilever Beam Design Problem'`

`class enoppy.paper_based.pdo_2022.CompressionSpringProblem(f_penalty=None)`

Bases: `enoppy.engineer.Engineer`

x = [x1, x2, x3, x4]

CSD aims to minimize the weight of a tension/compression spring given the values of 3 parameters: the wire diameter (d=x1), number of active coils (P=x3), and mean coil diameter (D=x2).

https://sci-hub.se/10.1016/s0166-3615(99)00046-9

`evaluate(x)`

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`

Compute the values of the constraint functions for a given set of input values.

`get_objs(x)`

Compute the values of the objective functions for a given set of input values.

`name = 'Compression Spring Design Problem'`

`class enoppy.paper_based.pdo_2022.CorrugatedBulkheadProblem(f_penalty=None)`

Bases: `enoppy.engineer.Engineer`

[x1, x2, x3, x4] = [width, depth, length, thickness]

`evaluate(x)`

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Corrugated Bulkhead Design Problem'

enoppy.paper_based.pdo_2022.**GTD**
alias of *enoppy.paper_based.pdo_2022.GearTrainProblem*

class enoppy.paper_based.pdo_2022.**GearTrainProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Unconstrained discrete design optimization problem [x1, x2, x3, x4] = [n_A, n_B, n_C, n_D]

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters x (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Gear Train Design Problem'

enoppy.paper_based.pdo_2022.**IBD**
alias of *enoppy.paper_based.pdo_2022.IBeamProblem*

class enoppy.paper_based.pdo_2022.**IBeamProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Minimizes the vertical deflection of a beam [x1, x2, x3, x4] = [b, h, t_w, t_f]

evaluate(*x*)
Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'I Beam Design Problem'

enoppy.paper_based.pdo_2022.**PLD**
alias of *enoppy.paper_based.pdo_2022.PistonLeverProblem*

enoppy.paper_based.pdo_2022.**PVP**
alias of *enoppy.paper_based.pdo_2022.PressureVesselProblem*

class enoppy.paper_based.pdo_2022.**PistonLeverProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

[x1, x2, x3, x4] = [H, B, D, X]

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Piston Lever Design Problem'

class enoppy.paper_based.pdo_2022.**PressureVesselProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4]

Variables: the inner radius (R=x3), the thickness of the head (Th=x2), the length of the cylindrical section of the vessel (L=x4), and the thickness of the shell (Ts=x1)

https://sci-hub.se/10.1115/1.2912596

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Pressure Vessel Design Problem'

enoppy.paper_based.pdo_2022.**RCB**
alias of *enoppy.paper_based.pdo_2022.ReinforcedConcreateBeamProblem*

class enoppy.paper_based.pdo_2022.**ReinforcedConcreateBeamProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

[x1, x2, x3]

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Reinforced Concreate Beam Design Problem'

enoppy.paper_based.pdo_2022.**SRD**
alias of *enoppy.paper_based.pdo_2022.SpeedReducerProblem*

class enoppy.paper_based.pdo_2022.**SpeedReducerProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Depicts a gearbox that sits between the propeller and engine of an aeroplane [x1, x2, x3, x4, x5, x6, x7] = [b, m, z, l1, l2, d1, d2]

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Speed Reducer Design Problem'

enoppy.paper_based.pdo_2022.**TBTD**
alias of *enoppy.paper_based.pdo_2022.ThreeBarTrussProblem*

enoppy.paper_based.pdo_2022.**TCD**
alias of *enoppy.paper_based.pdo_2022.TubularColumnProblem*

class enoppy.paper_based.pdo_2022.**ThreeBarTrussProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Minimize three-bar structure weight subject to supporting a total load P acting vertically downwards

[x1, x2]

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

`name = 'Three Bar Truss Design Problem'`

class enoppy.paper_based.pdo_2022.**TubularColumnProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

[x1, x2] = [d, t]

https://apmonitor.com/me575/index.php/Main/TubularColumn

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

`name = 'Tubular Column Design Problem'`

enoppy.paper_based.pdo_2022.**WBP**
alias of *enoppy.paper_based.pdo_2022.WeldedBeamProblem*

class enoppy.paper_based.pdo_2022.**WeldedBeamProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

x = [x1, x2, x3, x4]

WBD is subjected to 4 design constraints: shear, beam blending stress, bar buckling load beam, and deflection variables: h=x1, l=x2, t=x3, b=x4 l: length, h: height, t: thickness, b: weld thickness of the bar

https://sci-hub.se/10.1016/s0166-3615(99)00046-9

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Welded Beam Design Problem'

1.2.1.4 enoppy.paper_based.rwco_2020

enoppy.paper_based.rwco_2020.**BPSP**
alias of *enoppy.paper_based.rwco_2020.BlendingPoolingSeparationProblem*

class enoppy.paper_based.rwco_2020.**BlendingPoolingSeparationProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x37, x38] Blending-Pooling-Separation problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Blending-Pooling-Separation problem (Industrial Chemical Processes)'

enoppy.paper_based.rwco_2020.**CCSDP**
alias of *enoppy.paper_based.rwco_2020.TensionCompressionSpringDesignProblem*

enoppy.paper_based.rwco_2020.**HENDC1P**
alias of *enoppy.paper_based.rwco_2020.HeatExchangerNetworkDesignCase1Problem*

enoppy.paper_based.rwco_2020.**HENDC2P**
alias of *enoppy.paper_based.rwco_2020.HeatExchangerNetworkDesignCase2Problem*

enoppy.paper_based.rwco_2020.**HPP**
alias of *enoppy.paper_based.rwco_2020.HaverlyPoolingProblem*

class enoppy.paper_based.rwco_2020.**HaverlyPoolingProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x9] Haverly's Pooling Problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_eq_cons(x)`
Compute the values of the equality constraint functions for a given set of input values.

`get_ineq_cons(x)`
Compute the values of the inequality constraint functions for a given set of input values.

`get_objs(x)`
Maximum to minimum by using negative sign

`name = "Haverly's Pooling Problem (Industrial Chemical Processes)"`

class enoppy.paper_based.rwco_2020.**HeatExchangerNetworkDesignCase1Problem**(*f_penalty=None*)
Bases: `enoppy.engineer.Engineer`

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x9] Heat Exchanger Network Design (case 1)

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_eq_cons(x)`
Compute the values of the equality constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

`name = 'Heat Exchanger Network Design Case 1 (Industrial Chemical Processes)'`

class enoppy.paper_based.rwco_2020.**HeatExchangerNetworkDesignCase2Problem**(*f_penalty=None*)
Bases: `enoppy.engineer.Engineer`

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x10, x11] Heat Exchanger Network Design (case 2)

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_eq_cons(x)`
Compute the values of the equality constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

name = 'Heat Exchanger Network Design Case 2 (Industrial Chemical Processes)'

enoppy.paper_based.rwco_2020.**MDCBDP**
alias of *enoppy.paper_based.rwco_2020.MultipleDiskClutchBrakeDesignProblem*

enoppy.paper_based.rwco_2020.**MPBP**
alias of *enoppy.paper_based.rwco_2020.MultiProductBatchPlantProblem*

class enoppy.paper_based.rwco_2020.**MultiProductBatchPlantProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Process design and synthesis problems [x1, x2,. . . , x10] Multi-product batch plant

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Multi-product batch plant (Process design and synthesis problems)'

class enoppy.paper_based.rwco_2020.**MultipleDiskClutchBrakeDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3, x4, x5] Multiple disk clutch brake design problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Multiple disk clutch brake design problem (Mechanical design problems)'

enoppy.paper_based.rwco_2020.**OBJ11**(*x*, *n*)

enoppy.paper_based.rwco_2020.**ODIRSP**
alias of *enoppy.paper_based.rwco_2020.OptimalDesignIndustrialRefrigerationSystemProblem*

enoppy.paper_based.rwco_2020.**OOAUP**
alias of *enoppy.paper_based.rwco_2020.OptimalOperationAlkylationUnitProblem*

class enoppy.paper_based.rwco_2020.**OptimalDesignIndustrialRefrigerationSystemProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2,. . . , x14] Optimal design of industrial refrigeration system

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Optimal design of industrial refrigeration system (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**OptimalOperationAlkylationUnitProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x7] Optimal Operation of Alkylation Unit

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Optimal Operation of Alkylation Unit (Industrial Chemical Processes)'

enoppy.paper_based.rwco_2020.**PDP**
alias of *enoppy.paper_based.rwco_2020.ProcessDesignProblem*

enoppy.paper_based.rwco_2020.**PFSP**
alias of *enoppy.paper_based.rwco_2020.ProcessFlowSheetingProblem*

enoppy.paper_based.rwco_2020.**PGTDOP**
alias of *enoppy.paper_based.rwco_2020.PlanetaryGearTrainDesignOptimizationProblem*

enoppy.paper_based.rwco_2020.**PINBNSP**
alias of *enoppy.paper_based.rwco_2020.PropaneIsobutaneNButaneNonsharpSeparationProblem*

enoppy.paper_based.rwco_2020.**PS01P**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesis01Problem*

enoppy.paper_based.rwco_2020.**PS02P**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesis02Problem*

enoppy.paper_based.rwco_2020.**PSADP**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesisAndDesignProblem*

enoppy.paper_based.rwco_2020.**PVDP**
alias of *enoppy.paper_based.rwco_2020.PressureVesselDesignProblem*

class enoppy.paper_based.rwco_2020.**PlanetaryGearTrainDesignOptimizationProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2,. . . ,x9] Planetary gear train design optimization problem

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Planetary gear train design optimization problem (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**PressureVesselDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3, x4] Pressure vessel design

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Pressure vessel design (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**ProcessDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Process design and synthesis problems [x1, x2,. . . , x5] Process design Problem

amend_position(*x, lb=None, ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Process design Problem (Process design and synthesis problems)'

class enoppy.paper_based.rwco_2020.**ProcessFlowSheetingProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Process design and synthesis problems [x1, x2, x3] Process flow sheeting problem

amend_position(*x, lb=None, ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_ineq_cons(x)`
Compute the values of the inequality constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

`name = 'Process flow sheeting problem (Process design and synthesis problems)'`

`class enoppy.paper_based.rwco_2020.ProcessSynthesis01Problem(f_penalty=None)`
Bases: `enoppy.engineer.Engineer`

Process design and synthesis problems [x1, x2] Process synthesis problem 01

`amend_position(x, lb=None, ub=None)`
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

`get_ineq_cons(x)`
Compute the values of the inequality constraint functions for a given set of input values.

`get_objs(x)`
Compute the values of the objective functions for a given set of input values.

`name = 'Process synthesis 01 problem (Process design and synthesis problems)'`

`class enoppy.paper_based.rwco_2020.ProcessSynthesis02Problem(f_penalty=None)`
Bases: `enoppy.engineer.Engineer`

Process design and synthesis problems [x1, x2,. . . , x9] Process synthesis problem 02

`amend_position(x, lb=None, ub=None)`
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

`evaluate(x)`
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons(x)`
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Process synthesis 02 problem (Process design and synthesis problems)'

class enoppy.paper_based.rwco_2020.**ProcessSynthesisAndDesignProblem**(*f_penalty=None*)
Bases: `enoppy.engineer.Engineer`

Process design and synthesis problems [x1, x2, x3] Process synthesis and design problem

amend_position(*x*, *lb=None*, *ub=None*)
Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Process synthesis and design problem (Process design and synthesis problems)'

class enoppy.paper_based.rwco_2020.**PropaneIsobutaneNButaneNonsharpSeparationProblem**(*f_penalty=None*)
Bases: `enoppy.engineer.Engineer`

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x47, x48] Propane, Isobutane, n-Butane Nonsharp Separation

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Propane, Isobutane, n-Butane Nonsharp Separation (Industrial Chemical Processes)'

enoppy.paper_based.rwco_2020.**RNDP**
alias of *enoppy.paper_based.rwco_2020.ReactorNetworkDesignProblem*

class enoppy.paper_based.rwco_2020.**ReactorNetworkDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Industrial Chemical Processes [x1, x2, x3, x4,. . . , x6] Reactor Network Design Problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Reactor Network Design (Industrial Chemical Processes)'

class enoppy.paper_based.rwco_2020.**RobotGripperProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3, x4, x5] Robot gripper problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters x (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns val – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Robot gripper problem (Mechanical design problems)'

enoppy.paper_based.rwco_2020.**SCPP**
alias of *enoppy.paper_based.rwco_2020.StepConePulleyProblem*

class enoppy.paper_based.rwco_2020.**StepConePulleyProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3, x4, x5] Step-cone pulley problem

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)
Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Step-cone pulley problem (Mechanical design problems)'

enoppy.paper_based.rwco_2020.**TBTDP**
alias of *enoppy.paper_based.rwco_2020.ThreeBarTrussDesignProblem*

enoppy.paper_based.rwco_2020.**TRP**
alias of *enoppy.paper_based.rwco_2020.TwoReactorProblem*

class enoppy.paper_based.rwco_2020.**TensionCompressionSpringDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3] Tension/compression spring design

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Tension/compression spring design (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**ThreeBarTrussDesignProblem**(*f_penalty=None*)

Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2] Three-bar truss design problem

evaluate(*x*)

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)

Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Three-bar truss design problem (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**TwoReactorProblem**(*f_penalty=None*)

Bases: *enoppy.engineer.Engineer*

Process design and synthesis problems [x1, x2, . . . , x8] Two-reactor problem

amend_position(*x*, *lb=None*, *ub=None*)

Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

evaluate(*x*)

Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)

Compute the values of the constraint functions for a given set of input values.

get_eq_cons(*x*)

Compute the values of the equality constraint functions for a given set of input values.

get_ineq_cons(*x*)

Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)

Compute the values of the objective functions for a given set of input values.

name = 'Two-reactor problem (Process design and synthesis problems)'

enoppy.paper_based.rwco_2020.**WBDP**

alias of *enoppy.paper_based.rwco_2020.WeldedBeamDesignProblem*

enoppy.paper_based.rwco_2020.**WMSRP**

alias of *enoppy.paper_based.rwco_2020.WeightMinimizationSpeedReducerProblem*

class enoppy.paper_based.rwco_2020.**WeightMinimizationSpeedReducerProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2,. . . , x7] Weight minimization of a speed reducer

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Weight minimization of a speed reducer (Mechanical design problems)'

class enoppy.paper_based.rwco_2020.**WeldedBeamDesignProblem**(*f_penalty=None*)
Bases: *enoppy.engineer.Engineer*

Mechanical design problems [x1, x2, x3, x4] Welded beam design

evaluate(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

get_cons(*x*)
Compute the values of the constraint functions for a given set of input values.

get_ineq_cons(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

get_objs(*x*)
Compute the values of the objective functions for a given set of input values.

name = 'Welded beam design (Mechanical design problems)'

enoppy.paper_based.rwco_2020.**p1**
alias of *enoppy.paper_based.rwco_2020.HeatExchangerNetworkDesignCase1Problem*

enoppy.paper_based.rwco_2020.**p10**
alias of *enoppy.paper_based.rwco_2020.ProcessFlowSheetingProblem*

enoppy.paper_based.rwco_2020.**p11**
alias of *enoppy.paper_based.rwco_2020.TwoReactorProblem*

enoppy.paper_based.rwco_2020.**p12**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesis02Problem*

enoppy.paper_based.rwco_2020.**p13**
alias of *enoppy.paper_based.rwco_2020.ProcessDesignProblem*

enoppy.paper_based.rwco_2020.**p14**
alias of *enoppy.paper_based.rwco_2020.MultiProductBatchPlantProblem*

enoppy.paper_based.rwco_2020.**p15**
alias of *enoppy.paper_based.rwco_2020.WeightMinimizationSpeedReducerProblem*

enoppy.paper_based.rwco_2020.**p16**
alias of *enoppy.paper_based.rwco_2020.OptimalDesignIndustrialRefrigerationSystemProblem*

enoppy.paper_based.rwco_2020.**p17**
alias of *enoppy.paper_based.rwco_2020.TensionCompressionSpringDesignProblem*

enoppy.paper_based.rwco_2020.**p18**
alias of *enoppy.paper_based.rwco_2020.PressureVesselDesignProblem*

enoppy.paper_based.rwco_2020.**p19**
alias of *enoppy.paper_based.rwco_2020.WeldedBeamDesignProblem*

enoppy.paper_based.rwco_2020.**p2**
alias of *enoppy.paper_based.rwco_2020.HeatExchangerNetworkDesignCase2Problem*

enoppy.paper_based.rwco_2020.**p20**
alias of *enoppy.paper_based.rwco_2020.ThreeBarTrussDesignProblem*

enoppy.paper_based.rwco_2020.**p21**
alias of *enoppy.paper_based.rwco_2020.MultipleDiskClutchBrakeDesignProblem*

enoppy.paper_based.rwco_2020.**p22**
alias of *enoppy.paper_based.rwco_2020.PlanetaryGearTrainDesignOptimizationProblem*

enoppy.paper_based.rwco_2020.**p23**
alias of *enoppy.paper_based.rwco_2020.StepConePulleyProblem*

enoppy.paper_based.rwco_2020.**p3**
alias of *enoppy.paper_based.rwco_2020.HaverlyPoolingProblem*

enoppy.paper_based.rwco_2020.**p4**
alias of *enoppy.paper_based.rwco_2020.BlendingPoolingSeparationProblem*

enoppy.paper_based.rwco_2020.**p5**
alias of *enoppy.paper_based.rwco_2020.PropaneIsobutaneNButaneNonsharpSeparationProblem*

enoppy.paper_based.rwco_2020.**p6**
alias of *enoppy.paper_based.rwco_2020.OptimalOperationAlkylationUnitProblem*

enoppy.paper_based.rwco_2020.**p7**
alias of *enoppy.paper_based.rwco_2020.ReactorNetworkDesignProblem*

enoppy.paper_based.rwco_2020.**p8**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesis01Problem*

enoppy.paper_based.rwco_2020.**p9**
alias of *enoppy.paper_based.rwco_2020.ProcessSynthesisAndDesignProblem*

1.2.2 enoppy.utils

1.2.2.1 enoppy.utils.encoder

class enoppy.utils.encoder.**LabelEncoder**

Bases: object

Encode categorical features as integer labels.

fit(*y*)

Fit label encoder to a given set of labels.

y [array-like] Labels to encode.

fit_transform(*y*)

Fit label encoder and return encoded labels.

Parameters **y** (*array-like of shape (n_samples,)*) – Target values.

Returns **y** – Encoded labels.

Return type array-like of shape (n_samples,)

inverse_transform(*y*)

Transform integer labels to original labels.

y [array-like] Encoded integer labels.

original_labels [array-like] Original labels.

transform(*y*)

Transform labels to encoded integer labels.

y [array-like] Labels to encode.

encoded_labels [array-like] Encoded integer labels.

1.2.2.2 enoppy.utils.validator

enoppy.utils.validator.**check_bool**(*name: str*, *value: bool*, *bound=(True, False)*)

enoppy.utils.validator.**check_float**(*name: str*, *value: int*, *bound=None*)

enoppy.utils.validator.**check_int**(*name: str*, *value: int*, *bound=None*)

enoppy.utils.validator.**check_str**(*name: str*, *value: str*, *bound=None*)

enoppy.utils.validator.**check_tuple_float**(*name: str*, *values: tuple*, *bounds=None*)

enoppy.utils.validator.**check_tuple_int**(*name: str*, *values: tuple*, *bounds=None*)

enoppy.utils.validator.**is_in_bound**(*value*, *bound*)

enoppy.utils.validator.**is_str_in_list**(*value: str*, *my_list: list*)

1.2.3 enoppy.engineer

class enoppy.engineer.**Engineer**

Bases: `abc.ABC`

Defines an abstract class for engineering design problems.

All subclasses should implement the `evaluate` method for a particular problem.

bounds

The lower/upper bounds of the problem. This a 2D-matrix of [lower, upper] array that contain the lower and upper bounds. By default, each problem has its own bounds. But user can try to put different bounds to test the problem.

Type list

n_dims

The dimensionality of the problem. It is calculated from bounds

Type int

lb

The lower bounds for the problem

Type np.ndarray

ub

The upper bounds for the problem

Type np.ndarray

f_global

The global optimum of the evaluated function.

Type float

x_global

A list of vectors that provide the locations of the global minimum. Note that some problems have multiple global minima, not all of which may be listed.

Type np.ndarray

n_fe

The number of function evaluations that the object has been asked to calculate.

Type int

amend_position(*x*, *lb=None*, *ub=None*)

Amend position to fit the format of the problem

Parameters **x** (*np.ndarray*) – The current position (solution)

property bounds

The lower/upper bounds to be used for optimization problem. This a 2D-matrix of [lower, upper] array that contain the lower and upper bounds for the problem. The problem should not be asked for evaluation outside these bounds. `len(bounds) == n_dims`.

check_penalty_func(*func=None*)

check_solution(*x*)

Raise the error if the problem size is not equal to the solution length

Parameters **x** (*np.ndarray*) – The solution

convex = True

`create_solution()`
Create a random solution for the current problem

Returns **solution** – The random solution

Return type np.ndarray

`default_penalty`(*list_objs=None*, *list_cons=None*)

`differentiable = True`

`evaluate`(*x*)
Evaluation of the benchmark function.

Parameters **x** (*np.ndarray, list, tuple*) – The candidate vector for evaluating the benchmark problem. Must have `len(x) == self.n_dims`.

Returns **val** – the evaluated benchmark function

Return type float

`get_cons`(*x*)
Compute the values of the constraint functions for a given set of input values.

`get_eq_cons`(*x*)
Compute the values of the equality constraint functions for a given set of input values.

`get_ineq_cons`(*x*)
Compute the values of the inequality constraint functions for a given set of input values.

`get_objs`(*x*)
Compute the values of the objective functions for a given set of input values.

`get_paras()`
Return the parameters of the problem. Depended on function

`property lb`
The lower bounds for the problem

Returns **lb** – The lower bounds for the problem

Return type 1D-vector

`linear = False`

`property n_cons`
The number of constraint functions of the problem.

`property n_dims`
The dimensionality of the problem.

`property n_eq_cons`
The number of equality constraint functions of the problem.

`property n_ineq_cons`
The number of inequality constraint functions of the problem.

`property n_objs`
The number of objective functions of the problem.

`name = 'Benchmark name'`

`parametric = True`

`property ub`
The upper bounds for the problem

Returns **ub** – The upper bounds for the problem

Return type 1D-vector

1.3 Cite Us

If you are using enoppy in your project, we would appreciate citations:

```
@software{nguyen_van_thieu_2023_7953207,
  author       = {Nguyen Van Thieu},
  title        = {ENOPPY: A Python Library for Engineering Optimization Problems},
  month        = may,
  year         = 2023,
  publisher    = {Zenodo},
  doi          = {10.5281/zenodo.7953206},
  url          = {https://github.com/thieu1995/enoppy}
}

@article{van2023mealpy,
  title={MEALPY: An open-source library for latest meta-heuristic algorithms in Python},
  author={Van Thieu, Nguyen and Mirjalili, Seyedali},
  journal={Journal of Systems Architecture},
  year={2023},
  publisher={Elsevier},
  doi={10.1016/j.sysarc.2023.102871}
}
```

If you have an open-ended or a research question, you can contact me via nguyenthieu2102@gmail.com

1.4 Important links

- Official source code repo: https://github.com/thieu1995/enoppy
- Official document: https://enoppy.readthedocs.io/
- Download releases: https://pypi.org/project/enoppy/
- Issue tracker: https://github.com/thieu1995/enoppy/issues
- Notable changes log: https://github.com/thieu1995/enoppy/blob/master/ChangeLog.md
- Examples with different meapy version: https://github.com/thieu1995/enoppy/blob/master/EXAMPLES.md
- Join our telegram community: [link](https://t.me/+fRVCJGuGJg1mNDg1)
- **This project also related to my another projects which are "meta-heuristics" and "neural-network", check it here**
  - https://github.com/thieu1995/mealpy
  - https://github.com/thieu1995/permetrics
  - https://github.com/thieu1995/opfunu
  - https://github.com/thieu1995/metaheuristics
  - https://github.com/thieu1995/MetaCluster

- https://github.com/thieu1995/pfevaluator
- https://github.com/thieu1995/IntelELM
- https://github.com/thieu1995/MetaPerceptron
- https://github.com/thieu1995/GrafoRVFL
- https://github.com/thieu1995/reflame
- https://github.com/aiir-team

1.5 License

The project is licensed under GNU General Public License (GPL) V3 license.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X