Universiteit Leiden
The Netherlands

EXCERCISE SESSION:Desdeo

Course: Multicriteria Optimization and Decision Analysis

Instructor: Michael Emmerich

Presenter: Kamand Hajiaghapour

Sections

1

Introducing Desdeo

2

Solving a problem using Desdeo

Desdeo Introduction

- Decision Support for computationally Demanding Optimization problems
- DESDEO is a free and open-source Python-based framework
- DESDEO contains implementations of some interactive methods and modules
- Interactive methods are iterative by nature where a decision maker can direct the solution process

The interactive methods currently implemented in DESDEO

- The synchronous NIMBUS method
- Different variants of the NAUTILUS method family, including NAUTILUS Navigator
- Wierzbicki's reference point method
- Pareto navigator
- Interactive RVEA
- Interactive NSGA III

Solving a problem using Desdeo

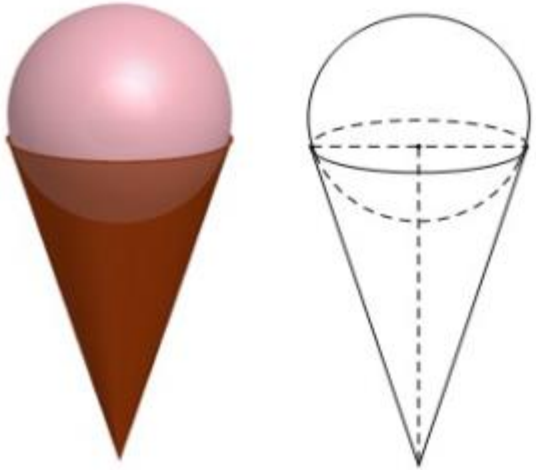
Problem description

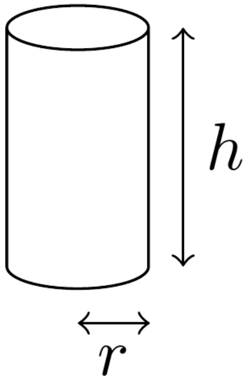
Optimizing geometrical shapes with desdeo

$$SurfaceArea(r,h) \rightarrow \min$$
$$Volume(r,h) \rightarrow \max$$

$r = radius$

h = height





area_icecreamcone = $\pi r\sqrt{(h^2 + r^2)}$ =np.pi * r * np.sqrt(h**2 + r**2)

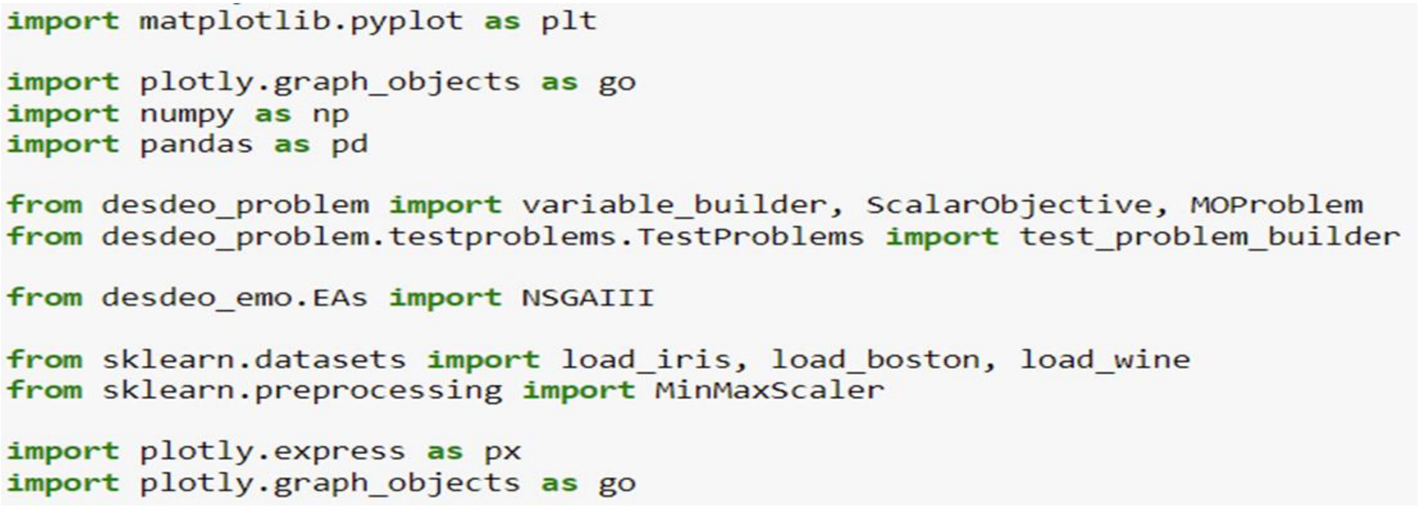
volume_filled_icecreamcone = $\frac{1}{3}\pi r^2 h + \frac{1}{2} * \frac{4}{3} * \pi r^3$ =(python) 1.0 /3.0 * (np.pi) * r**2 * h + 1/2*4.0 / 3.0 *np.pi *r**3

area_cylinder =$2\pi r^2 + 2\pi rh =$ (python) 2 * np.pi * r**2 + 2 * np.pi * r *h

volume_cylinder = $\pi r^2 h$ = (python) np.pi* r**2 * h

Importing necessary packages

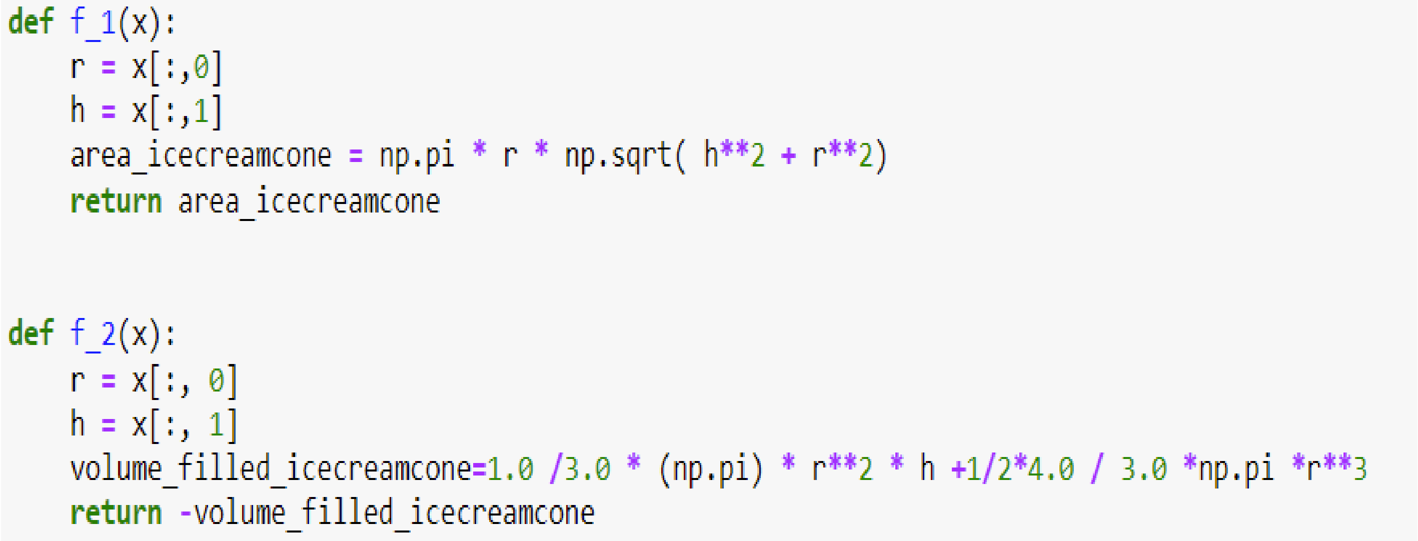
- Install desdeo_emo
- Before instantiating the problem instance, we have to create object to define each of the variables, objectives, and constraints.



```
import matplotlib.pyplot as plt

import plotly.graph_objects as go
import numpy as np
import pandas as pd

from desdeo_problem import variable_builder, ScalarObjective, MOProblem
from desdeo_problem.testproblems.TestProblems import test_problem_builder

from desdeo_emo.EAs import NSGAIII

from sklearn.datasets import load_iris, load_boston, load_wine
from sklearn.preprocessing import MinMaxScaler

import plotly.express as px
import plotly.graph_objects as go
```

Defining Objective Functions

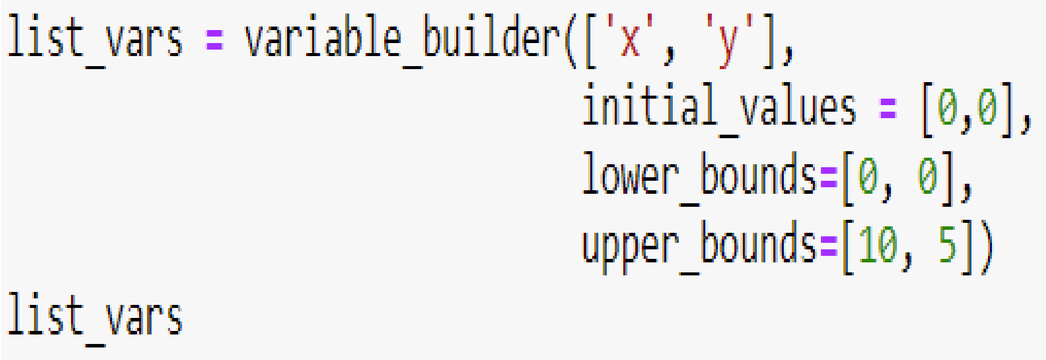
- Two objective functions:



```
def f_1(x):
    r = x[:,0]
    h = x[:,1]
    area_icecreamcone = np.pi * r * np.sqrt( h**2 + r**2)
    return area_icecreamcone


def f_2(x):
    r = x[:, 0]
    h = x[:, 1]
    volume_filled_icecreamcone=1.0 /3.0 * (np.pi) * r**2 * h +1/2*4.0 / 3.0 *np.pi *r**3
    return -volume_filled_icecreamcone
```

Creating Variable objects

- Two variables:
- Using Variable_builder function



```
list_vars = variable_builder(['x', 'y'],
                             initial_values = [0,0],
                             lower_bounds=[0, 0],
                             upper_bounds=[10, 5])
list_vars
```

Create Objective objects

To define an objective class instance, one needs to pass the following:

Objective name/s (Required)

Evaluator (Required for analytical/simulation based objectives)

Lower bound (Not required)

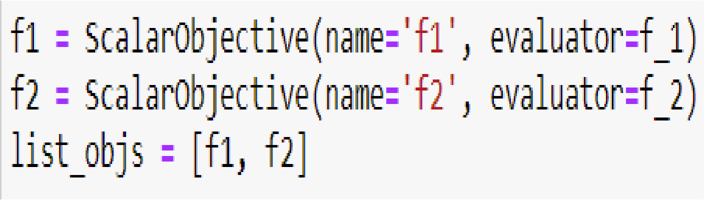
Upper bound (Not required)

maximize (Not required)

The DESDEO framework has the following classification for objectives, based on the kind of evaluater to be used:

“Scalar” objectives

“Vector” objectives



```
f1 = ScalarObjective(name='f1', evaluator=f_1)
f2 = ScalarObjective(name='f2', evaluator=f_2)
list_objs = [f1, f2]
```

Creating the problem object

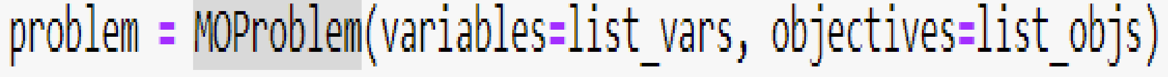
- Using MOProblem
- The output is a NamedTuple object:

objectives

fitness

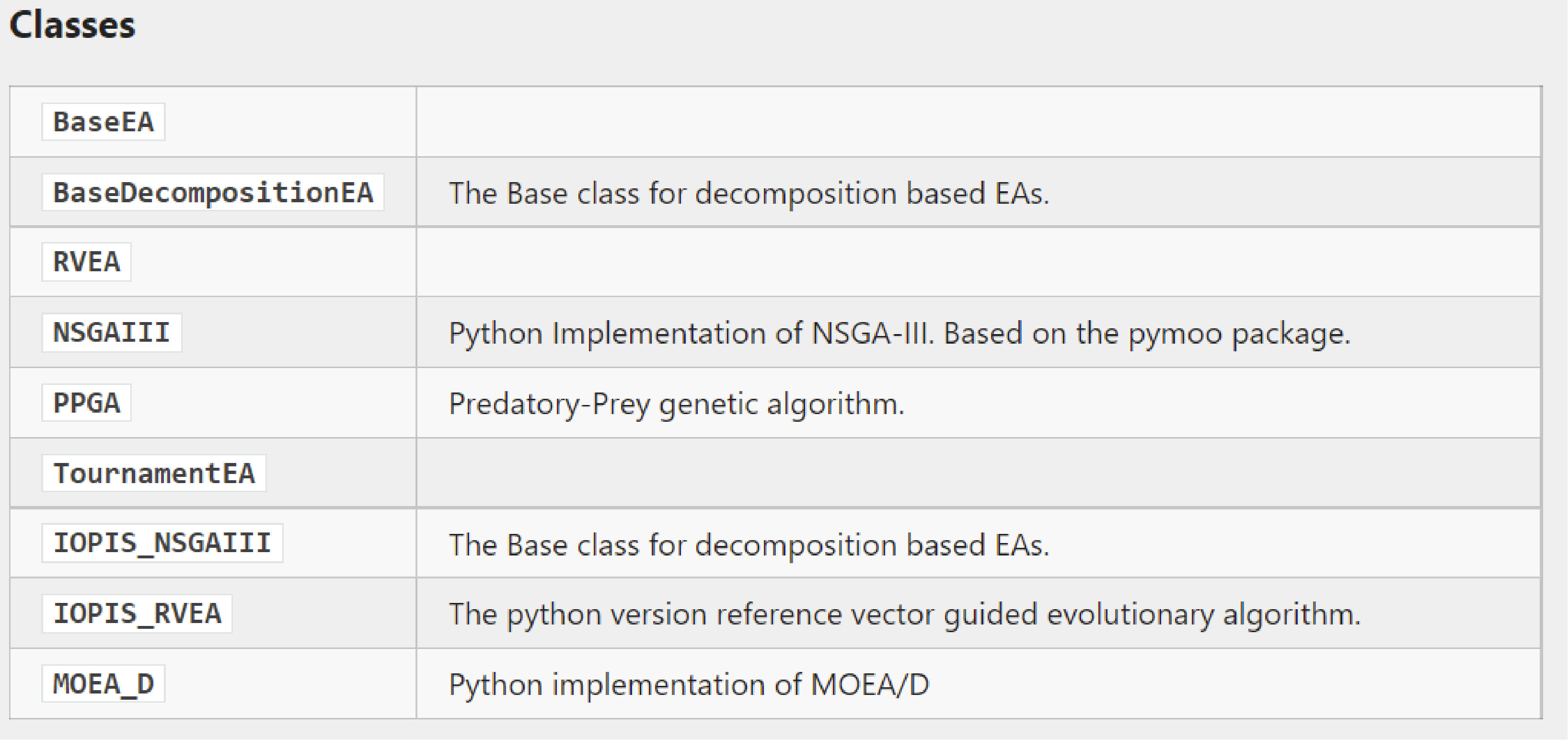
*constraints

uncertainity



```
problem = MOProblem(variables=list_vars, objectives=list_objs)
```

Using the Evolutionary Algorithms(Eas)



Classes

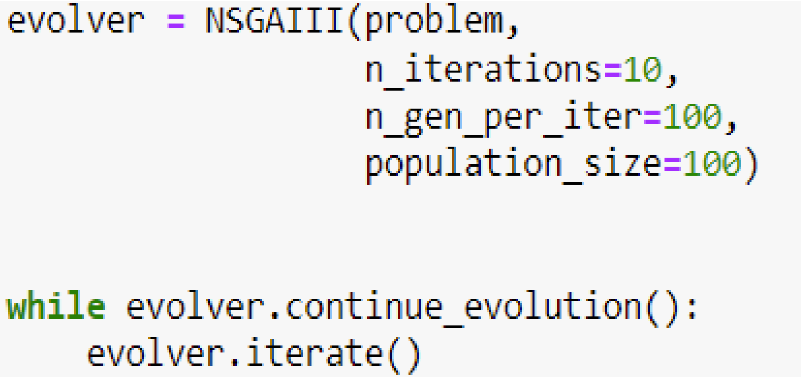
`BaseEA`	
`BaseDecompositionEA`	The Base class for decomposition based EAs.
`RVEA`	
`NSGAIII`	Python Implementation of NSGA-III. Based on the pymoo package.
`PPGA`	Predatory-Prey genetic algorithm.
`TournamentEA`	
`IOPIS_NSGAIII`	The Base class for decomposition based EAs.
`IOPIS_RVEA`	The python version reference vector guided evolutionary algorithm.
`MOEA_D`	Python implementation of MOEA/D

Using the EAs

Non-dominated Sorting Genetic Algorithm III

Parameters

- problem (MOProblem)
- population_size (int, optional)
- population_params (Dict, optional)
- initial_population (Population, optional)
- lattice_resolution (int, optional)
- selection_type (str, optional)
- a_priori (bool, optional)
- interact (bool, optional)
- n_iterations (int, optional)
- n_gen_per_iter (int, optional)
- total_function_evaluations (int, optional)

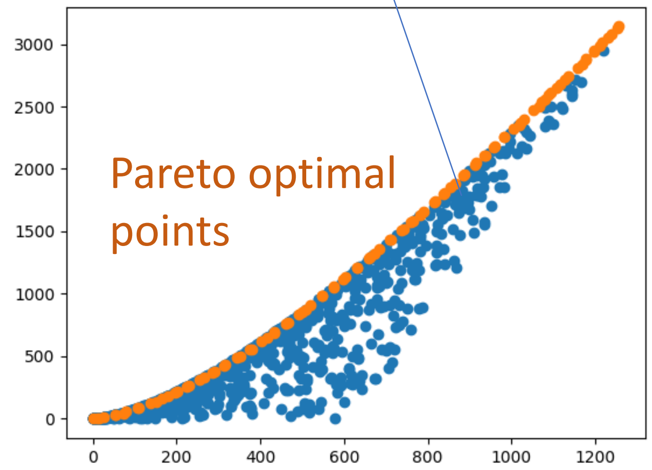


```
evolver = NSGAIII(problem,
                  n_iterations=10,
                  n_gen_per_iter=100,
                  population_size=100)


while evolver.continue_evolution():
    evolver.iterate()
```

Visualization

Random points

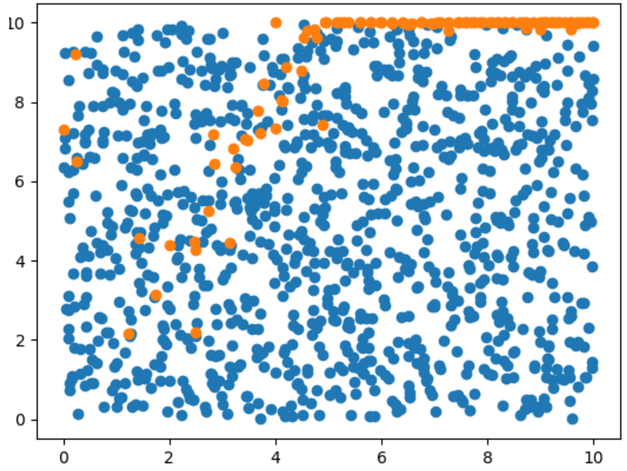
$V = f_2$



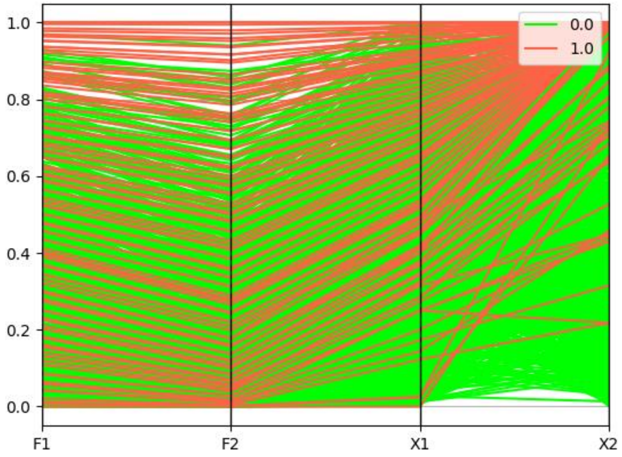
Pareto optimal points

$S = f_1$

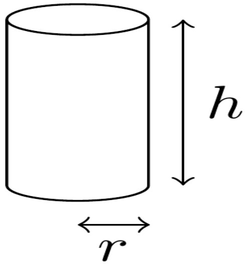
$h = x_2$



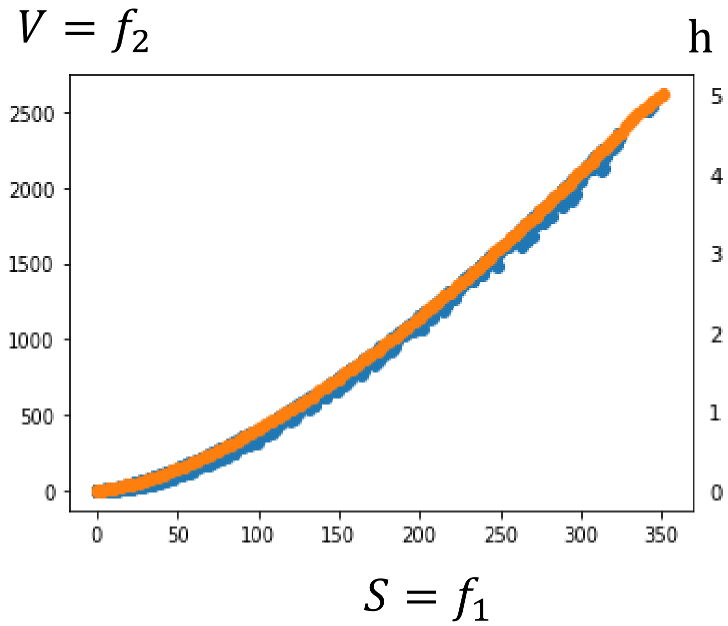
$r = x_1$

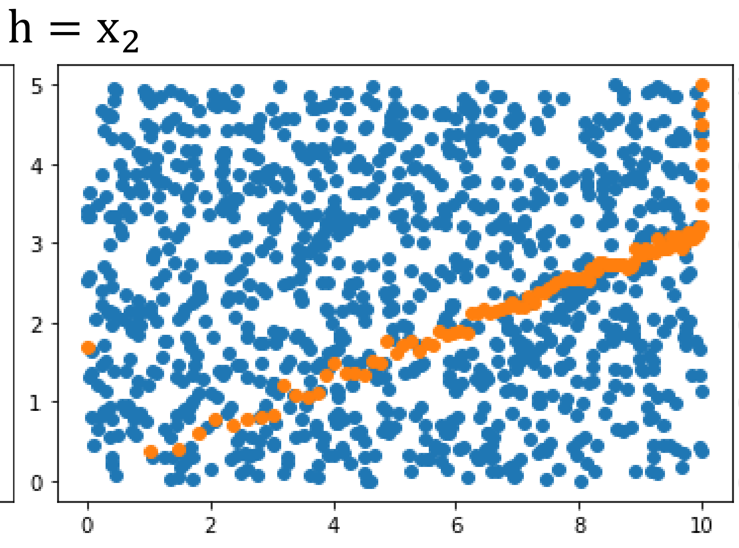


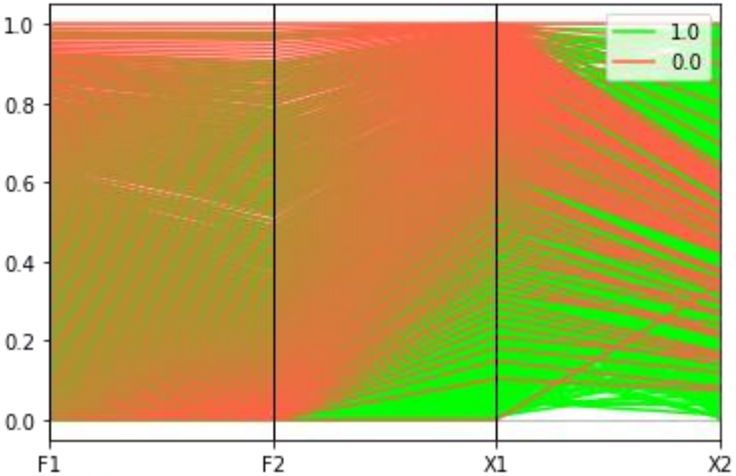
Parallel coordinates of concatenated and normalized set

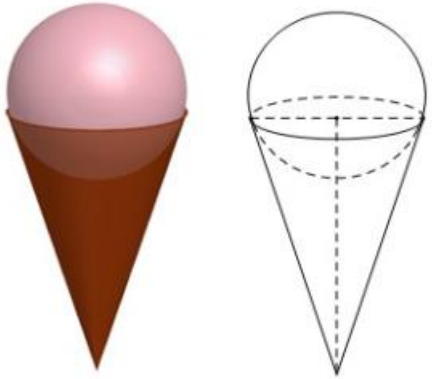


Visualization









References

- https://desdeo-problem.readthedocs.io/en/latest/notebooks/Defining_a_problem.html
- https://desdeo-emo.readthedocs.io/en/latest/autoapi/desdeo_emo/EAs/index.html

Thank you for your attention!