Thermea Project

Quentin Dumont

23 août 2021

Introduction

Presentation of the company

BDR THERMEA GROUP

- Dedicated to manufacture and sale of domestic and industrial heating appliances.
- Several brands : Baxi, De Dietrich, Remeha... (fusion in 2009)
- Head office in Apeldoorn (Netherlands)

Optimization

The subject : a thermodynamic Water-heater

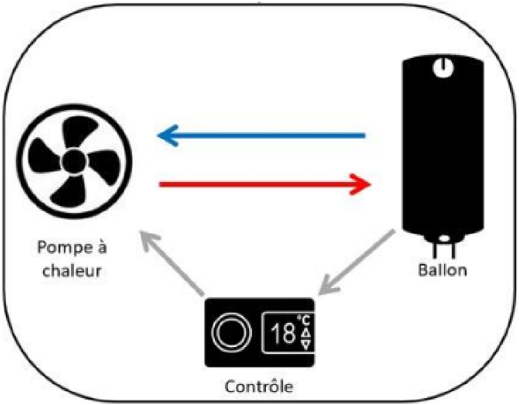


Figure – Thermo-Dynamical Water-Heater

Optimization

Several constraints of Optimization

Two criteria of performance of interest :

- COP_{DHW}
- $star\ notation$

Defined by :

- The european standards EN16147 norm [10]
- Specific other specifications like LCIE 103-15/C specifications [8].

Optimization

In order to solve this complex problem :

- $NSGA-II$
- $NSGA-III$
- etc..

Using the JmetalPy library [14]

Produc tested

Figure – IDU3001 [24]

Figure – TWH [25]

Charactestics

Charactestics :

- Compatible with the RT2012 [1] regulation and standard
- Compact : 560x586x1950 mm

Charactestics :

- Plenty of domestic hot water (214 -270 litres)
- hot water heating up to 62°C with PAC
- Dimension : 610x610x1690 mm

1. Focusing on the new market when building new homes to meet ecological requirements

Dymola

Dymola [21]

A complete tool for modeling and simulation of complex systems used in many areas as robotic, aeronautics, thermodynamic, ...

FMU

After the models (IDU3FS and TWH3001) are built by Dymola, we can use a Python library called **FMpy** [33] in order to speed up the simulation and to launch it in our code.

FMpy [33]

FMpy is a free Python library to simulate **Functional Mock-up Units**(FMUs) that :

- supports FMI 1.0 and 2.0
- supports Co-Simulation and Model exchange
- has a command line, graphical user interface

Parameters

Parameters

3 parameters are indispensable to permit the proper functioning of the simulation : T_{set},$\Delta T_{hysteresis}$ and H_{gauge}

- T_{set} is the maximal temperature to reach
- $\Delta T_{hysteresis} = T_{set}$ -T_{min} with T_{min} the minimal temperature to reach
- H_{gauge} is the height-location (from the top) of the temperarature sensor

Objectives

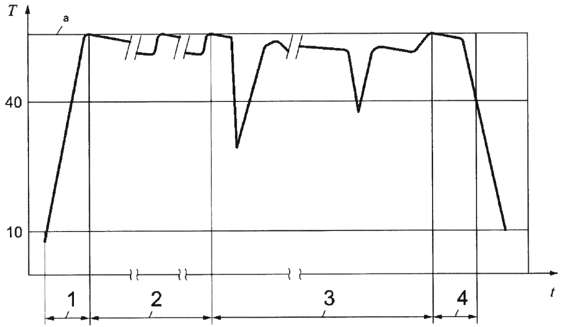
Evaluated in real conditions :

- to obtain COP_{DHW}
- to value its *star notation*

the test :

- requires different specific steps
- evaluates the quality of the products

Figure – EN16147 norm [10]



Légende

1	[Étape C] remplissage et période de mise en température (voir 7.7)	T	température
2	[Étape D] Puissance absorbée en régime stabilisé (voir 7.8)	t	temps
3	[Étape E] Puisages d'eau (voir 7.9)	a	température de consigne
4	[Étape F] Eau mitigée à 40 °C et température d'eau chaude de référence (voir 7.10)		

Thanks to the FMU, A very large amount of information (outputs) is computed.
These information contain different values as :

- COP_{DHW}
- V_{40}
- P_{es}
- θ'_{wh}
- t_h
- $CPUtime$
- ...

Figure – LCIE 103-15 [27]

Grandeur mesurée	Abréviation	Unité	Catégorie ★★	Catégorie ★★★
Capacité de stockage	Vm	l	$\geq V_n$	$\geq V_n$
Température d'eau chaude de référence	θ'_{WH}	°C	≥ 52,5	≥ 52,5
Puissance absorbée en régime stabilisé	P_{es}	kW	$\leq 0.0001 * V_n + 0.029 + (20 - \theta_{as})/1000$	$\leq 0.0001 * V_n + 0.024 + (20 - \theta_{as})/1000$
Charge thermique de l'appoint électrique		W/cm²	≤ 12	≤ 12
Volume d'eau mitigée à 40°C	V_{40}	l	$\geq (\theta_A - 10) / 30 / 1.33 * V$	$\geq (\theta_A - 10) / 30 / 1.22 * V$
Efficacité énergétique	η_{WH}	%	$\geq Qref / (Qref + 2.44) + \theta_{sc} / 100$	$\geq Qref / (Qref + 1.95) + \theta_{sc} / 100$
Durée de mise en température : Air extrait, air extrait mélangé, air extrait multisource Autres technologies	t_h	*h.min*	 ≤ 18.00 ≤ 14.00	 ≤ 18.00 ≤ 14.00
Enclenchement de l'appoint électrique (si existant) [4]			Enclenchement possible durant les étapes C,D, E ou F	Enclenchement non autorisé durant les étapes C,D, E ou F

By convention, 1 star notation : passing the EN16147 norm.

We work on a discretization of the possible attributes, we define the paramaters' range and design space by a cubic-grid.

Cubic-grid

Cubic-grid has $N = 16 \times 29 \times 19 = 8816$ possible designs [a], as :

- $T_{set} \in [315.15, 330.15]$ every 1 Kelvin degree, as $dim(\mathbb{T}_{set}) = 16$
- $\Delta T_{hysteresis} \in [2, 30]$ every 1 degree, as $dim(\Delta\mathbb{T}_{hysteresis}) = 29$
- $H_{gauge} \in [0, 0.9]$ every 0.05 meter, as $dim(\mathbb{H}_{gauge}) = 19$

a. Some of these designs are not physically possible

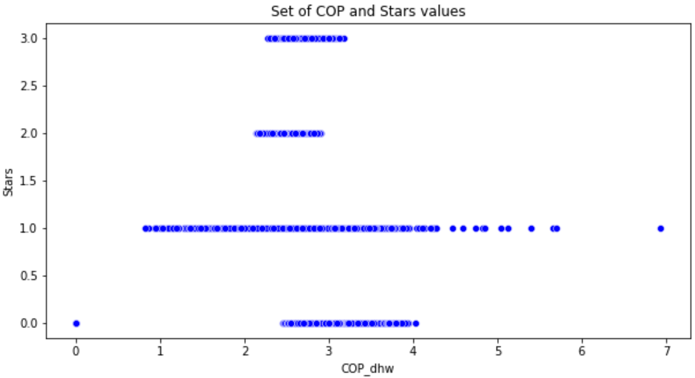
formulation

The main optimization problem can be formulated as such :

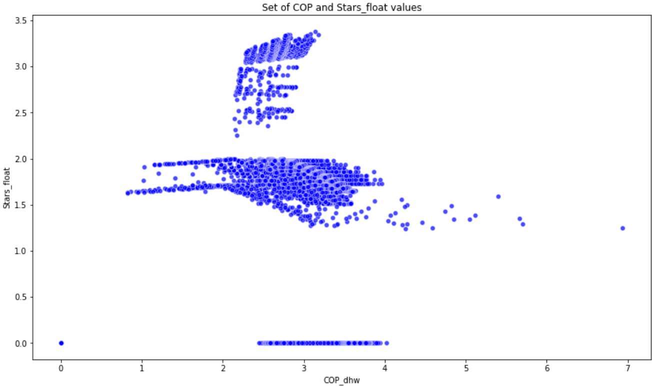
$$\underset{u \in \mathcal{X}}{\operatorname{argmax}}(COP_{DHW}(u), stars(u))$$

where $\mathcal{X} = \mathbb{T}_{set} \times \Delta\mathbb{T}_{hysteresis} \times \mathbb{H}_{gauge}$

Data


Set of COP and Stars values
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Stars
0
1
2
3
4
5
6
7
COP_dhw

In this way, we obtain this graphic :



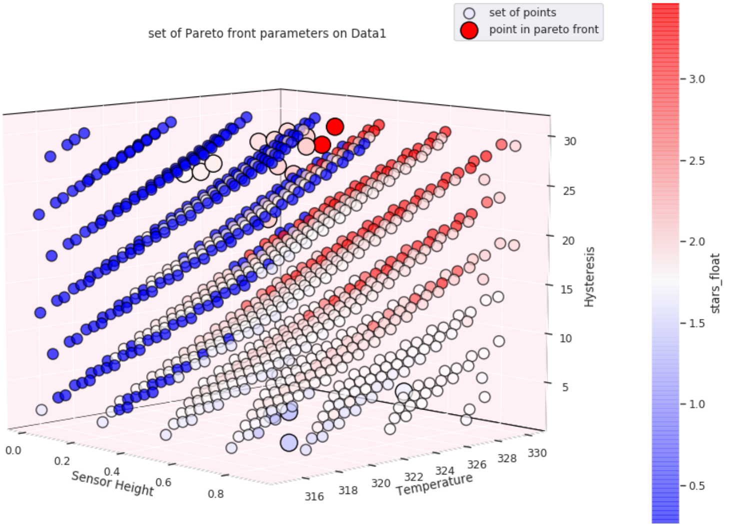


Figure – COP_{DHW} value of IDU3FS [24] depending on T_{set}, $\Delta T_{hysteresis}$ and H_{gauge}

Thermea Project
Quentin Dumont
Introduction
Project : Optimization of a Water heater
Project reformulation
IDU3FS
TWH3001
Dymola and FMU
Paramaters, objectives and data
Metaheuristic algorithms
Evaluating parameters and algorithms
Statistics
Bibliography

Multi-objective optimization

Multi-objective optimization problems deals with conflicting objectives. This comes from the fact that we cannot say which is "best".

Here is an example :

We are interested in the following mathematical problem :

- f_m an function of objective ($COP(x)$ and $stars(x)$ for example)
- $x = (x_1, x_2, .., x_n)^T$ a vector

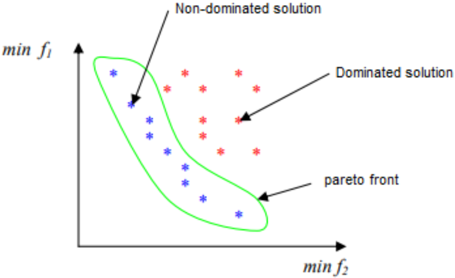
$$min_{x \in K}(f_m(x)), \quad m \in \{1, 2\}, \quad K\ compact$$

with constraints :

$$g_j(x) \geq 0, \quad j = 1, 2, ...n_1$$

and illustrated by :

Figure – Pareto front[17]



Domination

A solution x_i dominates x_j if both condition 1 and 2 below are true :

- Condition 1 : x_i is no worse than x_j for all objectives
- Condition 2 : x_i is strictly better than x_j in at least one objective

Mathematical notation : $x_j \preceq x_i$

Non-dominated solution

Among a set of solutions M, the non-dominated solutions are those that are not dominated by any member of this set. The others are so called "dominated solution"

Pareto front

the non-dominated set of solutions is called "pareto front"

Figure – Pareto front of the datafile number 1

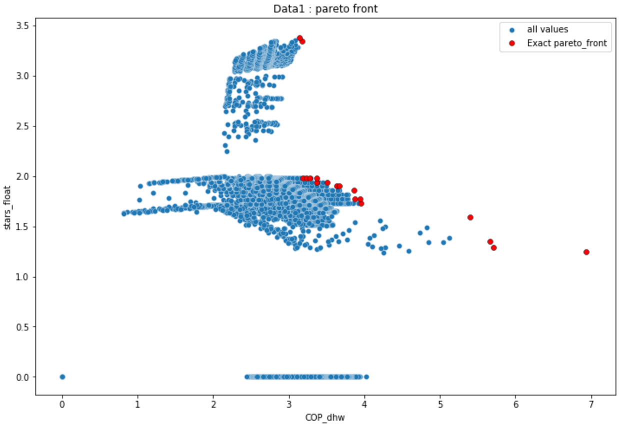


Figure – Pareto front of the datafile number 2

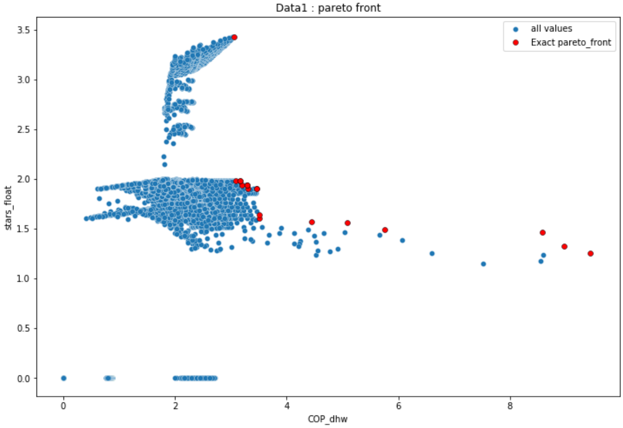
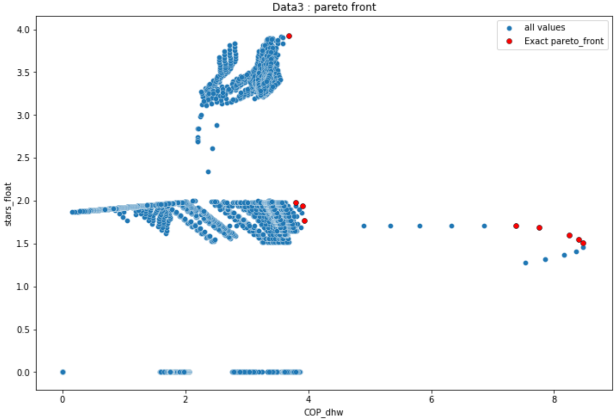
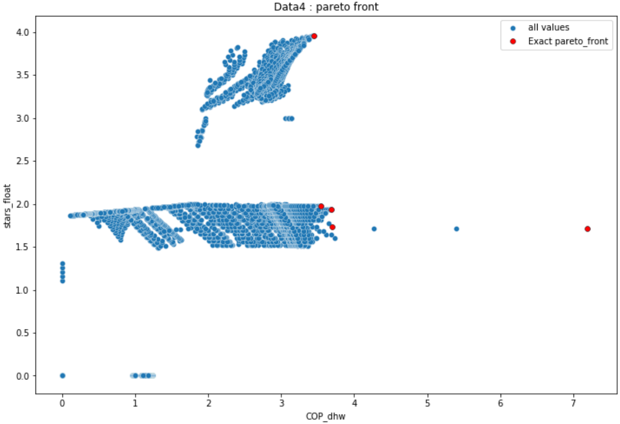


Figure – Pareto front of the datafile number 3



Thermea Project
Quentin Dumont
Introduction
Project : Optimization of a Water heater
Project reformulation
IDU3FS
TWH3001
Dymola and FMU
Paramaters, objectives and data
Metaheuristic algorithms
Evaluating parameters and algorithms
Statistics
Bibliography

Figure – Pareto front of the datafile number 4



Metaheuristic algorithms

It is not doable to explore all possibilites in a reasonable amount of time

Main idea

The main idea behind a genetic algorithm is to consider the set of all possible parameters as a population of individuals, each with their own attributes (the parameters) and fitness (objectives). An individual is considered to be fitter than an other individual if he has better objectives.

In our case :

A set given of attributes $(x_1, x_2, x_3) \in \mathbb{T}_{set} \times \Delta\mathbb{T}_{hysteresis} \times \mathbb{H}_{gauge}$ is viewed as an individual with attributes (x_1, x_2, x_3) and the corresponding couple $(COP_{DHW}, stars\_float)$ value is its fitness.

We will focus on particular two families of metaheuristic algorithms : **Genetic algorithms**[26] and **Particle swarm optimization**[29]

Genetic algorithms

- Popularised by John H. Holland [12] from 1975
- Inspired from the theory of evolution, process of natural Selection
- Involve randomness
- Useful to solve complex problem

Particle swarm optimization

Particle swarm optimization is one of the most well-known methaeuristic optimization technique based on swarm, which was proposed by Eberhart and Kennedy[31] from 1995.

Main idea

This algorithms inspired from swarm behavior such as bird flocking in nature.
It simulates animal's social behavior and cooperative way to find food, and each member in the swarms keeps changing the search pattern according to the learning experiences of its own and other members.

Implementation

JmetalPy

- Done by the JmetalPy's library [14]
- An object-oriented Python-base framework for multi-objective optimization with metaheuristic techniques
- Created in 2006 by Antonio Benitez-Hidalgo, Antonio J.Nebro, José Garcia-Nieto, ..

Why ?

- A full redesign from scratch in 2015
- An easy use of parallel computing
- A large amount of metaheuristic algorithms.

4 "candidates" of metaheuristic algorithms (2 **genetic algorithms** and 2 **particle swarm optimization's algorithms**) and 1 algorithm of random search (**our comparative algorithm**) in JmetalPy.
These 4 "candidates" are as follows :

- $NSGA - II$ (genetic-algorithm)
- $NSGA - III$ (genetic-algorithm)
- $MOPSO$ (particle swarm optimization)
- $SMPSO$ (particle swarm optimization)

Our motivation is twofold. First, we want to compare each these multi-objective optimization's algorithms consisting in analysing the performance. Secondly, we want to study the convergence speed with the idea of number of individuals computed.

Termination criterion

We opted for Inverted Generational Distance (IGD) as a criteria to stop our algorithm. This measure use the true pareto front as a reference and compare each elements of the true pareto front with the pareto front returned by our algorithm.
This measure is defined by :

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

where n is the number of elements in the true pareto front and d_i, the euclidean distance between the true pareto front's points and the nearest points of the pareto front found by our algorithm.
When $IGD = 0$, all the points generated by our algorithm are in the true pareto front.

Goal

The objective is to find the algorithm and its parameters that converge quicker to that best known individual.

Each algorithm need a specific set of parameters to be runned :

- NSGA2 : paramater p = [problem, seeds, mut_pbs, index_pbs , n_inits, cx_pbs]
- NSGA3 : parameter p = [problem, seeds, mut_pbs, index_pbs , n_inits, cx_pbs]
- MOPSO : parameter p = [problem, seeds, mut_pbs, perturbation,n_inits]
- SMPSO : parameter p = [problem , seeds, mut_pbs, index_pbs, n_inits]

parameters

- `problem` : a set of 1 problem
- `seeds` : a set of 10 different seeds
- `datafiles` : a set of 1 datafile to run the algorithms on
- `n_inits` :`n_inits` $= \{50, 150, 250\}$
- `cx_pb` : `cx_pbs` $= \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$
- `mut_pb` : `mut_pbs` $= \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$
- `index_pb` : `ind_pbs` $= \{10, 30, 50, 200, 300, 500, 1000, 50000\}$
- `perturbation` : `perturbation` $= \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$

This would make for each algorithms, a total of :

- NSGA2 : $10 \times 1 \times 11 \times 8 \times 3 \times 11 = 29040$ executions
- NSGA3 : $10 \times 1 \times 11 \times 8 \times 3 \times 11 = 29040$ executions
- MOPSO : $10 \times 1 \times 11 \times 1 \times 3 \times 6 = 1980$ executions
- SMPSO : $10 \times 1 \times 11 \times 8 \times 3 = 2640$ executions

This step allows to do a pre-treatment to know an interval of the best parameters of each algorithms. We observed that :

- index_pbs not have much influence on the sucess rate [2] of the convergence for $NSGA2$ and $NSGA3$
- the rate sucess is better when n_inits = 250 for $NSGA2$ and $NSGA3$ or n_inits = 50, 150 for $MOPSO$ and $SMPSO$

2. Sucess rate : convergence with an IGD ≤ 0.001

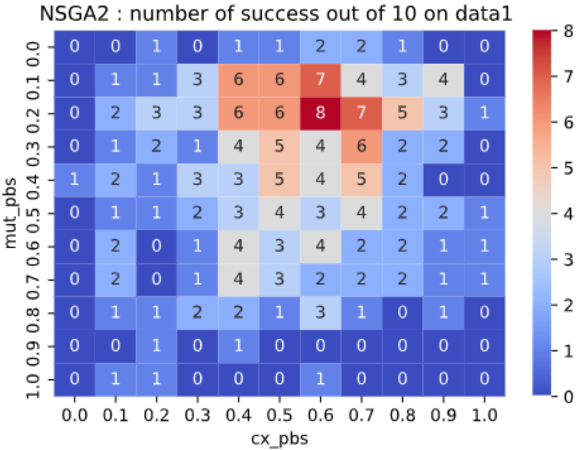


Figure – NSGA2 : number of success out of 10 on data1

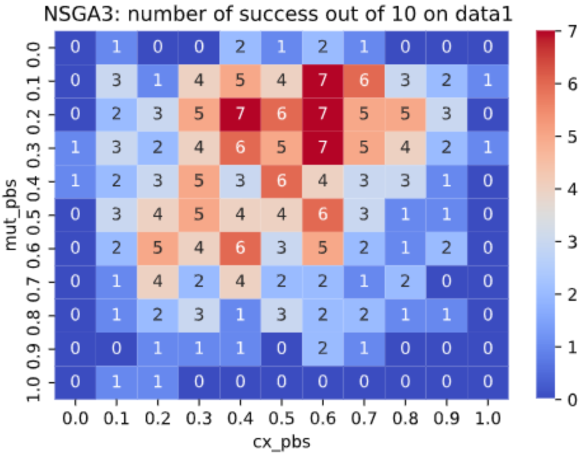


Figure – NSGA3 : number of success out of 10 on data1

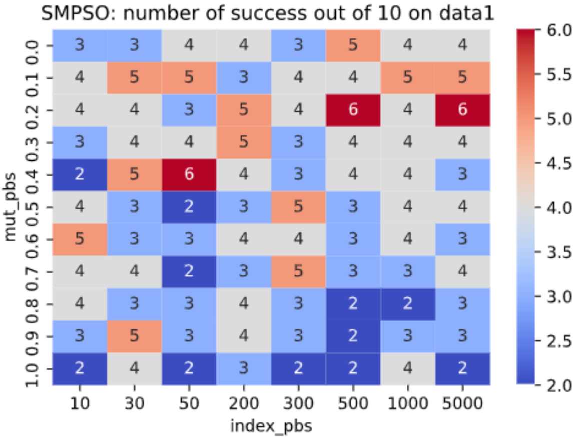


Figure – SMPSO : number of success out of 10 on data1

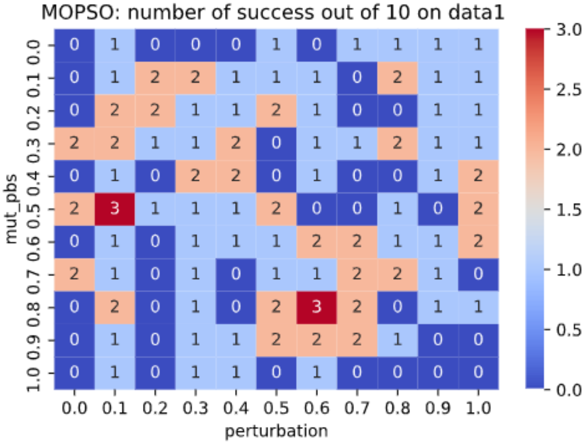


Figure – MOPSO : number of success out of 10 on data1

We can now reduce considerably the number of execution's algorithms by selecting :

- `problem` : a set of 1 problem
- `seeds` : a set of 100 different seeds
- `datafiles` : a set of 1-4 datafiles to run the algorithms on
- `n_inits` :`n_inits` $= \{50, 150\}$ (PSO) and $\{250\}$ (NSGA)
- `cx_pb` : `cx_pbs` $= \{0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.70\}$
- `mut_pb` : `mut_pbs` $= \{0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$ (NSGA)
- `mut_pb` : `mut_pbs` $= \{0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6\}$ (PSO)
- `index_pb` : `ind_pbs` $= \{10, 30, 50, 200, 300, 500, 1000, 50000\}$ (PSO) or $\{30\}$ (NSGA)

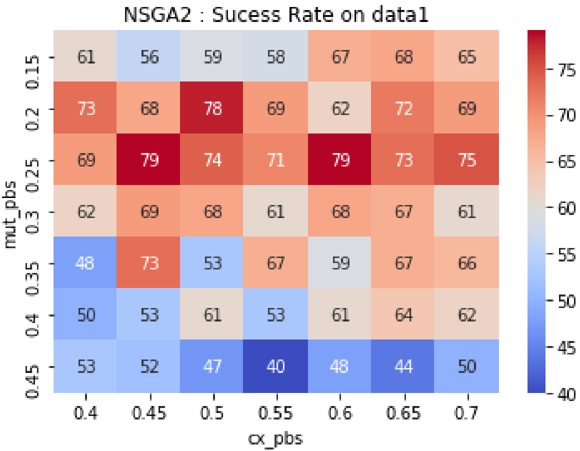


Figure – Sucess rate with NSGA2 on the file 1

Figure – Sucess rate with NSGA3 on the file 1

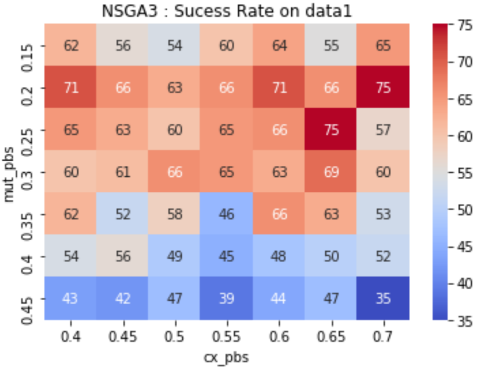
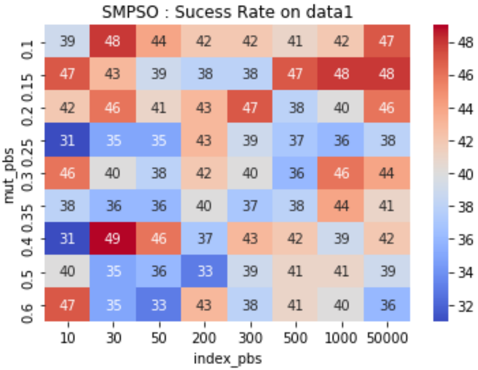


Figure – Sucess rate with SMPSO on the file 1



We focus on the number of point found of the true pareto front in the following way :

- Data number 1 has 18 points on the true front pareto, so we want to find at least 17 points
- Data number 2 has 18 points on the true front pareto, so we want to find at least 17 points
- Data number 3 has 9 points on the true front pareto, so we want to find at least 8 points
- Data number 4 has 5 points on the true front pareto, so we want to find at least 3 points

Figure – Sucess rate with NSGA2 on the file 1

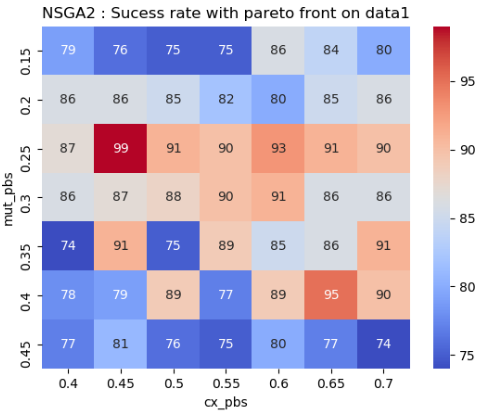


Figure – Sucess rate with NSGA2 on the file 2

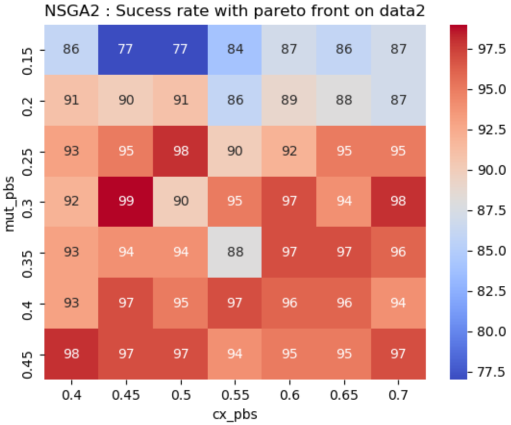


Figure – Sucess rate with NSGA2 on the file 3

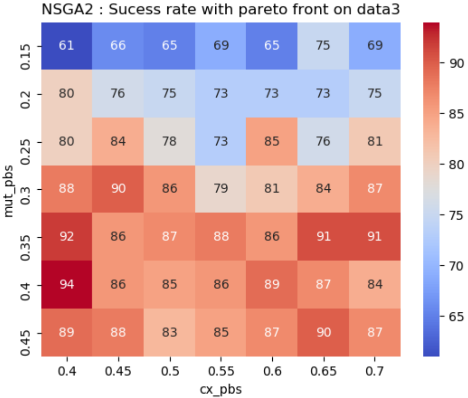


Figure – Sucess rate with NSGA2 on the file 4

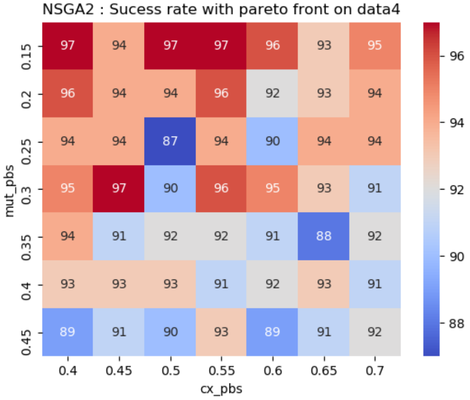


Figure – Sucess rate with NSGA3 on the file 1

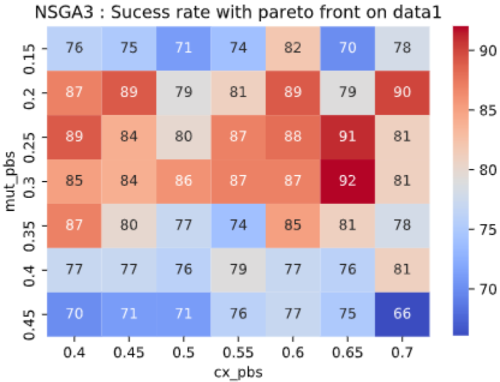


Figure – Sucess rate with NSGA3 on the file 2

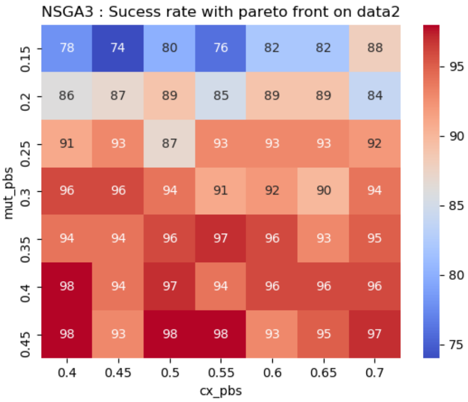


Figure – Sucess rate with NSGA3 on the file 3

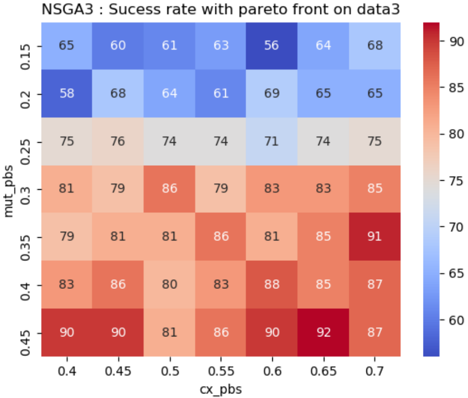


Figure – Sucess rate with NSGA3 on the file 4

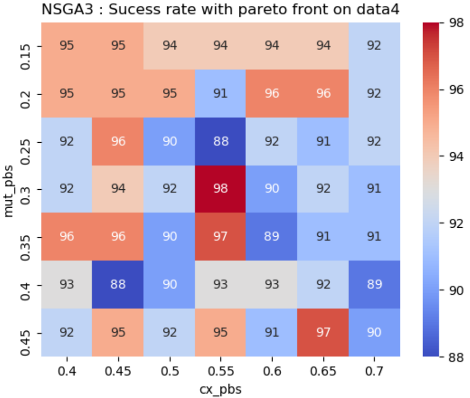


Figure – Sucess rate with SMPSO on the file 1

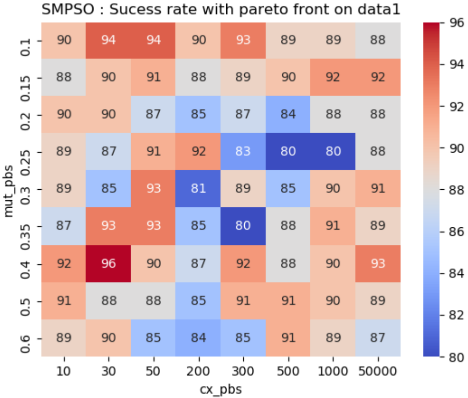


Figure – Sucess rate with SMPSO on the file 2

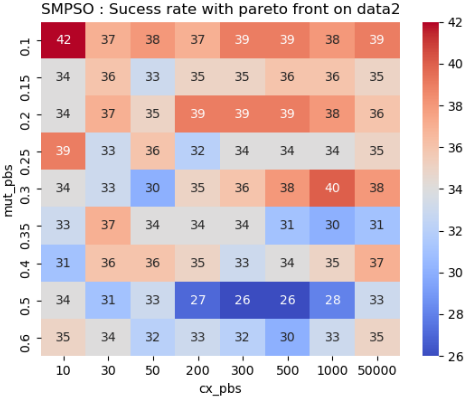


Figure – Sucess rate with SMPSO on the file 3

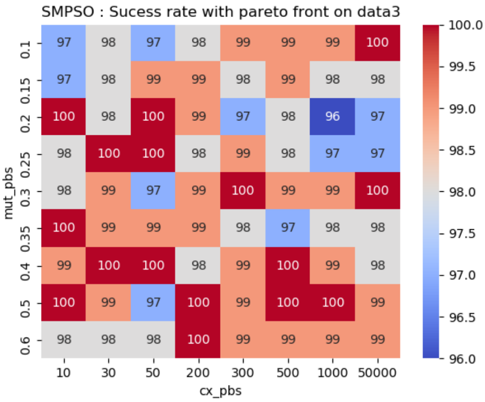
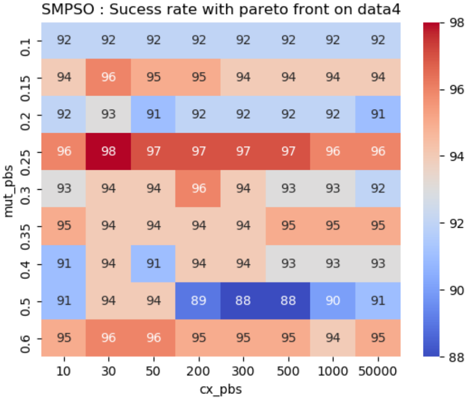


Figure – Sucess rate with SMPSO on the file 4



Classification

- $p = [\texttt{cx\_pb},\ \texttt{mut\_pb},\ \texttt{ind\_pb}]$
- The number of unique individuals
- we can store the quantity $\texttt{unique\_evals}(\texttt{alg},\ \texttt{datafile},\ p,\ \texttt{seed})$
- In the end, we store a quantile (90%) of `unique_evals`

We can use this notation : $q_{90}(\texttt{alg},\ \texttt{datafile},\ p)$ (this value depends on the set of seeds that we used).

Choosing the overall best set of parameters for an algoritm I

Three different options to decide on the overall best parameter p for the algorithms `alg` :

- "minmax" classification
- "points" classification
- "sucess rate" classification

Choosing the overall best set of parameters for an algoritm II

minmax classification

For each set of parameters, we store its worst performances over all the datafiles

$$q_{90}(\text{alg},\ \text{datafile}_{i_{\textbf{worst}}},\ p) = \max_{i=1,2} q_{90}(\text{alg},\ \text{datafile}_i,\ p)$$

we take the parameter $p_{j_{best}}$

$$p_{j_{\textbf{best}}} = \min_{j=1,\ldots 175} \left(\max_{i=1,2} q_{90}(\text{alg},\ \text{datafile}_i,\ p_j) \right)$$

Second option : a "points" classification

"points" classification

For each data file `datafile`, we classify the sets of parameters from the one with the smallest (best) quantile to the one with the biggest (worst) quantile. With this ranking, each set of parameters gets a number of points equal to its position in the list (a high quantile is worth more points).
We then add up the points of each set of parameters accross all the data files to get a final ranking, and we choose the one which got the fewest points.

Third option : a "sucess rate" classification

"sucess rate" classification

For each data file `datafile`, we count the convergence's sucess rate of each sets of parameters as follows :

- α = sucess rate of n points of the true pareto front found
- β = sucess rate of n-1 points of the true pareto front found

and the score become :

$$score = 2\alpha + \beta$$

We then add up the score of each set of parameters accross all the data files to get a final ranking, and we choose the one which got the fewest score.

NSGA2 parameters	TWH-L	TWH-M	IDU3FS-L	IDU3FS-M	minmax	points	sucess rate
$p_1 = (0.25, 0.6)$	4^{st}	7^{st}	2^{st}	23^{st}	$1^{st}(3552)$	1^{st}	10^{st}
$p_2 = (0.3, 0.4)$	5^{st}	14^{st}	17^{st}	20^{st}	$2^{st}(3640)$	3^{st}	11^{st}
$p_3 = (0.3, 0.5)$	12^{st}	9^{st}	1^{st}	8^{st}	$3^{st}(3768)$	2^{st}	9^{st}
$p_4 = (0.3, 0.45)$	16^{st}	15^{st}	3^{st}	22^{st}	$4^{st}(3805)$	3^{st}	8^{st}
$p_5 = (0.4, 0.65)$	22^{st}	26^{st}	27^{st}	41^{st}	$10^{st}(3864)$	9^{st}	1^{st}

FIGURE 39 – Best parameters for NSGA2 (notation p_i) through various classifications

NSGA3 parameters	TWH-L	TWH-M	IDU3FS-L	IDU3FS-M	minmax	points	sucess rate
$p_1 = (0.3, 0.5)$	22^{st}	1^{st}	4^{st}	17^{st}	$1^{st}(3923)$	1^{st}	1^{st}

FIGURE 40 – Best parameters for NSGA3 (notation p_i) through various classifications

SMPSO parameters	TWH-L	TWH-M	IDU3FS-L	IDU3FS-M	minmax	points	sucess rate
$p_1 = (0.35, 30, 50)$	16^{st}	13^{st}	121^{st}	83^{st}	$1^{st}(3766)$	12^{st}	5^{st}
$p_2 = (0.25, 500, 50)$	26^{st}	15^{st}	59^{st}	56^{st}	$2^{st}(3774)$	3^{st}	8^{st}
$p_3 = (0.25, 200, 50)$	29^{st}	14^{st}	96^{st}	56^{st}	$3^{st}(3786)$	8^{st}	7^{st}
$p_4 = (0.25, 1000, 50)$	20^{st}	18^{st}	45^{st}	44^{st}	$4^{st}(3791)$	1^{st}	6^{st}
$p_5 = (0.25, 5000, 50)$	14^{st}	20^{st}	46^{st}	50^{st}	$5^{st}(3795)$	2^{st}	1^{st}

FIGURE 41 – Best parameters for SMPSO (notation p_i) through various classifications

Application : classification on datafiles

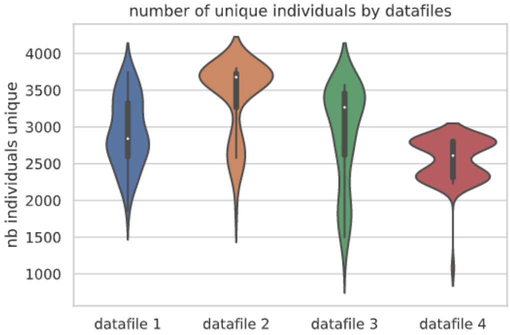


Figure – NSGA2 : violinplots for each datafiles

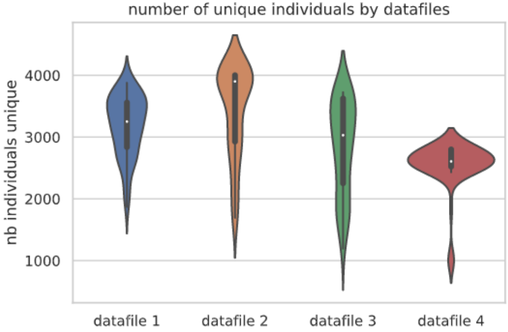


Figure – NSGA3 : violinplots for each datafiles

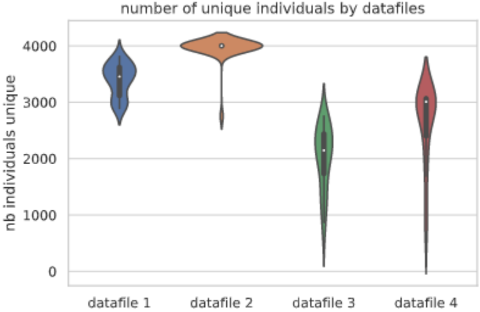


Figure – SMPSO : violinplots for each datafiles

Randomsearch

Randomsearch

The random search follows the hypergeometric law :

$$X \sim \mathrm{H}(N, n, p)$$

with N : populatin size, n : sample size, p : the probability of the issue.
And :

$$P(X = k) = \frac{\binom{Np}{k}\binom{N(1-p)}{n-k}}{\binom{N}{n}}$$

In our case :

- N_i : population size for each datafile
- n_i : 4000 for each datafile
- p_i : probability to found k point of the true pareto front out of total points k^i_{total}

k varies from 1 to k^i_{total} for each datafiles

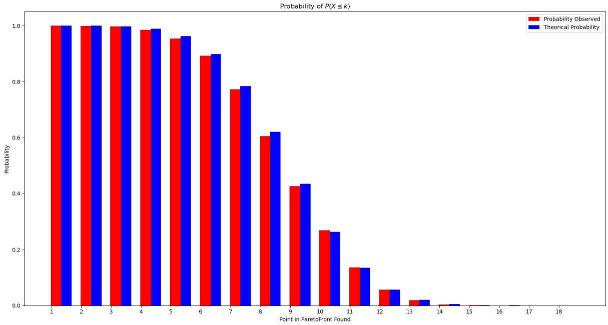


Figure – Randomsearch on IDU3FS cycle L

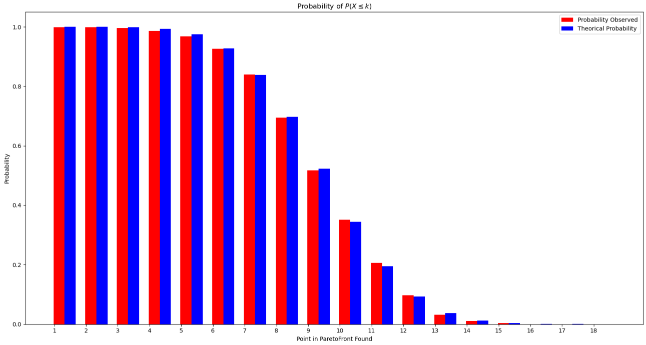


Figure – Randomsearch on IDU3FS cycle M

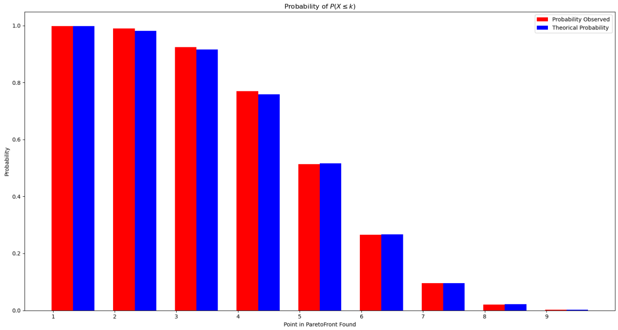


Figure – Randomsearch on THW3001 cycle L

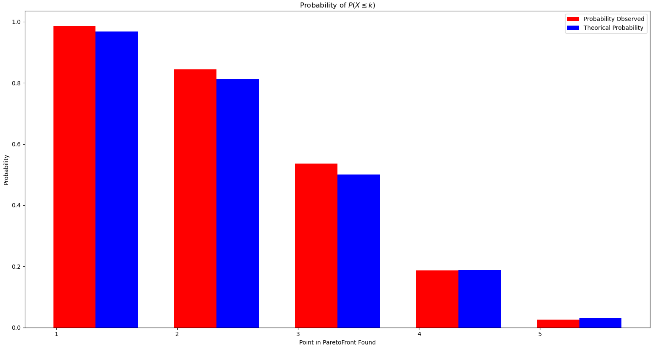


Figure – Randomsearch on THW3001 cycle M

Comparaison with our algorithm

We make a comparaison between Randomsearch and our 2 genetic algorithms.

The following graphics show the probability observed to found $k^i_{total} - 1$ [3] for each datafiles :

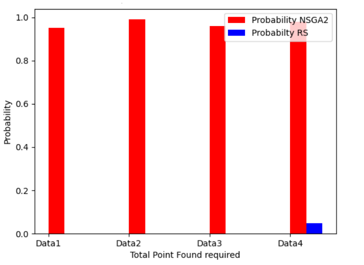


Figure – $NSGA2 : P(X \geq k^i_{itotal} - 1)$

Thermea Project
Quentin Dumont
Introduction
Project : Optimization of a Water heater
Project reformulation
IDU3FS
TWH3001
Dymola and FMU
Paramaters, objectives and data
Metaheuristic algorithms
Evaluating parameters and algorithms
Statistics
Bibliography

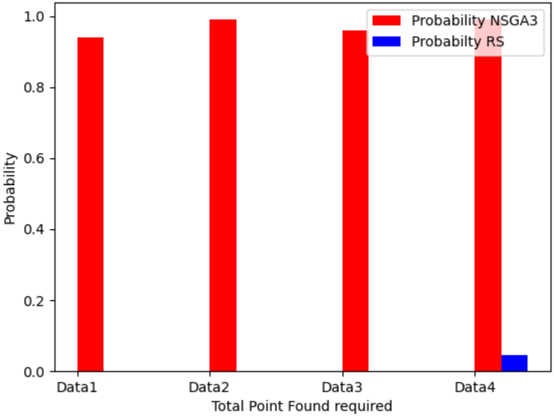
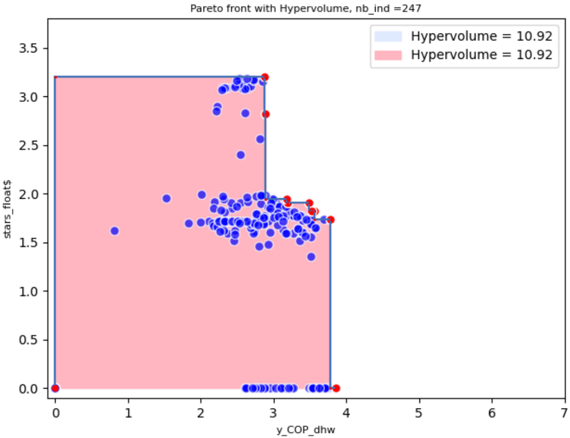


Figure – $NSGA3 : P(X \geq k^{i}_{total} - 1)$


Pareto front with Hypervolume, nb_ind =247
Hypervolume = 10.92
Hypervolume = 10.92
stars_float$
y_COP_dhw
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7

Link : https://youtu.be/TtzC31BGj9k

Conclusion

Conclusion

In conclusion to this project, we have realized that metaheuristic algorithms are a promising method for solving complex optimization problems such as the maximization of the $(COP_{DHW}, Stars\_float)$. They allowed us to find a solution by exploring roughly a quarter of the design space and win a lot of time for the simulations.

References I

- https://deap.readthedocs.io/en/master/tutorials/basic/part1.html
- https://irma-web1.math.unistra.fr/ vigon/liens/
- https://www.dedietrich-thermique.fr/la-marque/presse-recrutement/presse-professionnelle
- https://ebmpapst.atlassian.net/wiki/spaces/PS/pages/8716290/
- https://stackoverflow.com/questions/3594750/how-to-import-a-dll-function-in-c
- https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/
- library deap
- pareto
- chauffe eau

References II

- NORM EN16147
- https://www.dedietrich-thermique.fr/
- Holland
- https://pymoo.org/algorithms/nsga2.html
- https://github.com/jMetal/jMetalPy/tree/master/jmetal
- https://deap.readthedocs.io/en/master/api/tools.html
- https://deap.readthedocs.io/en/master/examples/
- https://tel.archives-ouvertes.fr/tel-00959099/document
- https://github.com/DEAP/deap/blob/master/examples/ga/nsga2.py

References III

- https://engineering.purdue.edu/ sudhoff/ee630/Lecture04.pdf
- NSGA2 publication
- https://www.mathworks.com/products/connections/product_detail/dymola.html
- https://pymoo.org/algorithms/nsga3.html
- https://en.wikipedia.org/wiki/Metaheuristic
- https://www.dedietrich-thermique.fr/nos-produits/pompe-a-chaleur/strateo
- https://www.dedietrich-thermique.fr/nos-produits/chauffe-eau-thermodynamique/kaliko
- Genetic algorithms
- star criteria

References IV

- https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac
- https://www.sciencedirect.com/topics/engineering/particle-swarm-optimization
- https://www.intechopen.com/chapters/69586
- https://ieeexplore.ieee.org/document/488968
- https://www.intechopen.com/chapters/69586
- https://github.com/CATIA-Systems/FMPy
- reference point
- https://arxiv.org/pdf/1903.02915.pdf
- Crowding Distance