# Introduction to Optimization

September 16, 2016

TC2 - Optimisation

Université Paris-Saclay, Orsay, France

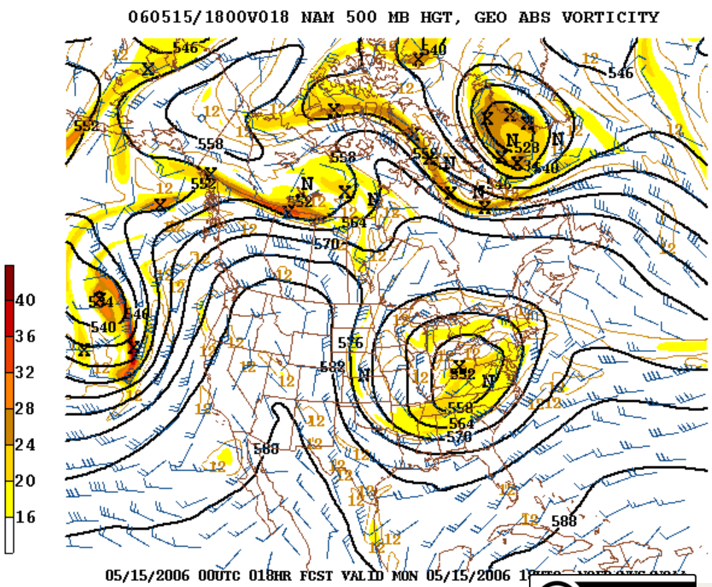
Anne Auger

Inria Saclay – Ile-de-France

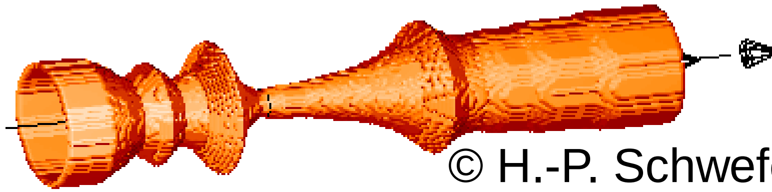
Dimo Brockhoff

Inria Saclay – Ile-de-France

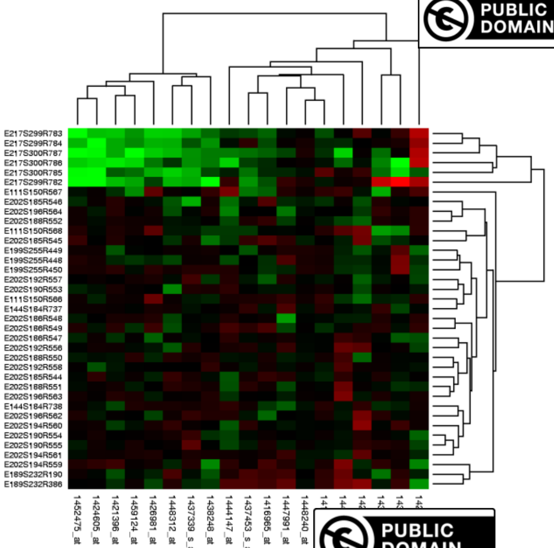
# What is Optimization?



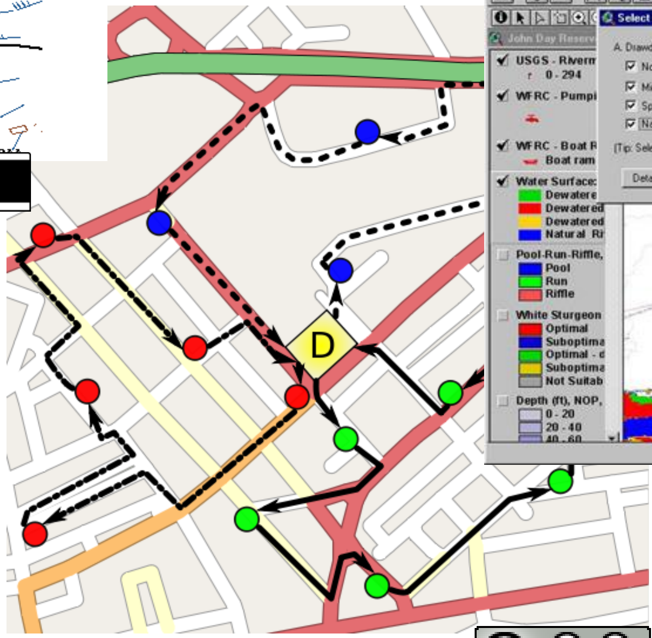


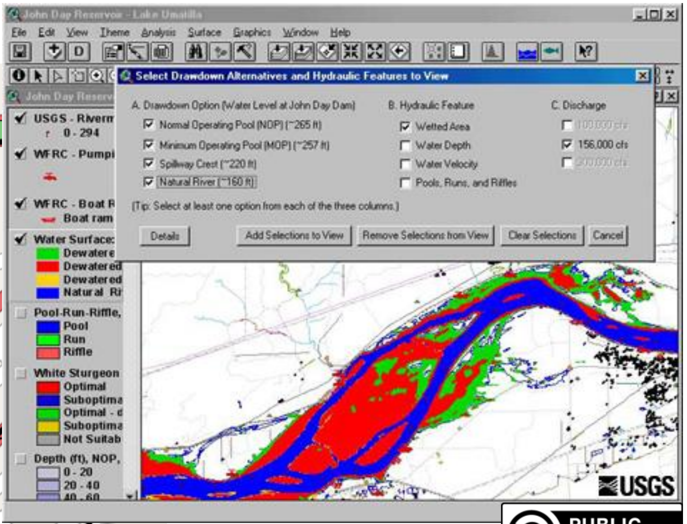



© H.-P. Schwefel










Maly LOLek

# What is Optimization?

Typically, we aim at

- finding solutions x which minimize f(x) in the shortest time possible (maximization is reformulated as minimization)
- or finding solutions x with as small f(x) in the shortest time possible (if finding the exact optimum is not possible)

# Course Overview

| Date | | Topic |
|---|---|---|
| Fri, 16.9.2016 | DB | Introduction and Group Project |
| Fri, 23.9.2016 | DB | Benchmarking with the COCO Platform (Group Project) |
| Fri, 30.9.2016 | AA | Introduction to Continuous Optimization |
| Fri, 7.10.2016 | AA | Gradient-Based Algorithms |
| Fri, 14.10.2016 | DB | Graph Theory, Greedy Algorithms and Branch and Bound |
| Tue, 18.10.2016 | DB | Dynamic programming, Approximation Algorithms and Heuristics |
| vacation | | |
| Fri, 4.11.2016 | AA | Stochastic Algorithms and Derivative-free Optimization |
| 14 - 18.11.2016 | | Exam (exact date to be confirmed) |

all classes + exam are from 14h till 17h15 (incl. a 15min break)
here in E107/D103, except for Tue, 18.10.2016 (E212/D103)

# Remarks

- possibly not clear yet what the lecture is about in detail
- but there will be always examples and small exercises to learn "on-the-fly" the concepts and fundamentals

**Overall goals:**

❶ give a broad overview of where and how optimization is used

❷ understand the fundamental concepts of optimization algorithms

❸ be able to apply common optimization algorithms on real-life (engineering) problems

there will be also an optional class "Blackbox Optimization" which we will present at the end of this lecture

# The Exam

- open book: take as much material as you want
- (most likely) multiple-choice
- date to be confirmed soon, but within November 14–18, 2016

- counts 2/3 of overall grade

# Group Project (aka "contrôle continu")

- we will have one group project with 4-5 students per group
- counts as 1/3 of overall grade
- the basic ideas: each group...
  - reads a scientific paper about an optimization algorithm
  - implements this algorithm
  - connects it to the benchmarking platform COCO
  - runs the algorithm with COCO to produce benchmarking data
  - compares their algorithm with others

# Group Project: Grading

- counts as 1/3 of overall grade
- grading mainly based on
  - a technical report (10 pages) to be handed in by October 21
  - an oral (group) presentation in the week November 7-11
- grading partly based on
  - each student's contribution to the group (via a written document to be signed by each student)
  - the online documentation (in a provided wiki)
  - the submitted source code
  - the timely submission of all required documents

# Course Overview

| | | |
|---|---|---|
| 1 | Fri, 16.9.2016 | Today's lecture: more infos in the end |
| | Wed, 21.9.2016<br>Thu, 22.9.2016 | groups defined via wiki<br>everybody went (actively!) through the Getting Started part of github.com/numbbo/coco |
| 2<br><br><br>3<br>4 | Fri, 23.9.2016<br><br><br>Fri, 30.9.2016<br>Fri, 7.10.2016 | Lecture, final adjustments of groups<br>everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture)<br>Lecture<br>Lecture |
| | Mon, 10.10.2016 | deadline for intermediate wiki report:<br>what has been done and what remains to be done? |
| 5<br>6 | Fri, 14.10.2016<br>Tue, 18.10.2016 | Lecture<br>Lecture |
| | Tue, 18.10.2016<br>Fri, 21.10.2016 | deadline for submitting data sets<br>deadline for paper submission |
| <br>7 | <br>Fri, 4.11.2016 | vacation<br>Final lecture |
| | 7.-11.11.2016 | oral presentations (individual time slots) |
| | 14 - 18.11.2016 | Exam (exact date to be confirmed) |

All deadlines:
23:59pm Paris time

# Group Project (aka “contrôle continu”)

- more detailed information in the end of today's lecture

All information also available at

`http://researchers.lille.inria.fr/~brockhof/optimizationSaclay/`

(group project info + link to wiki, lecture slides, ...)

# Overview of Today's Lecture

- More examples of optimization problems
  - introduce some basic concepts of optimization problems such as domain, constraint, ...
- Beginning of continuous optimization part
  - typical difficulties in continuous optimization
  - basics of benchmarking blackbox optimization algorithms with the COCO platform
  - basics needed for group project (more next week)

# General Context Optimization

**Given:**

set of possible solutions — *Search space*

quality criterion — *Objective function*

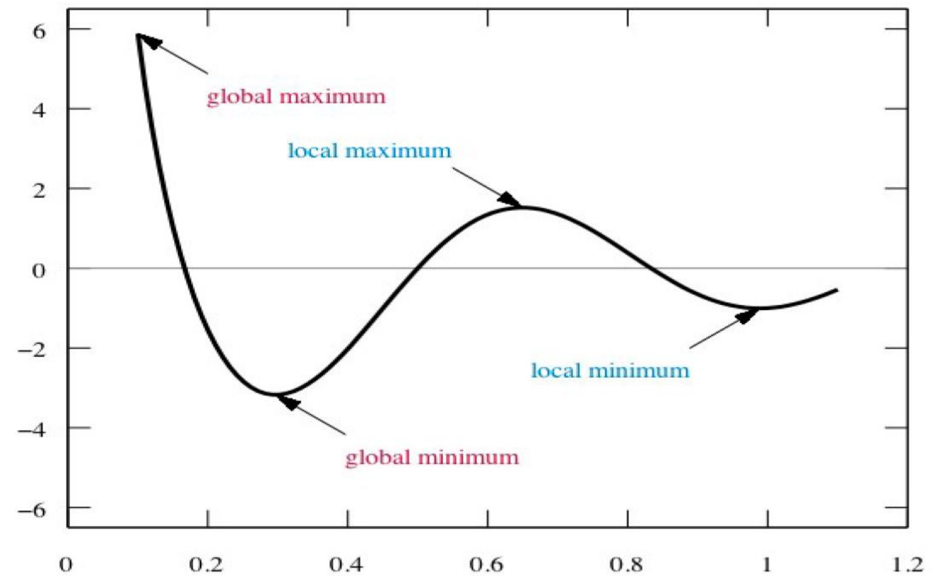
**Objective:**

Find the best possible solution for the given criterion

**Formally:**

Maximize or minimize

$$\mathcal{F} : \Omega \mapsto \mathbb{R},$$
$$x \mapsto \mathcal{F}(x)$$



# Constraints

Maximize or minimize

$$\mathcal{F} : \Omega \mapsto \mathbb{R},$$
$$x \mapsto \mathcal{F}(x)$$

unconstrained $\Omega$

Maximize or minimize

$$\mathcal{F} : \Omega \mapsto \mathbb{R},$$
$$x \mapsto \mathcal{F}(x)$$
$$\text{where} \quad g_i(x) \leq 0$$
$$h_j(x) = 0$$

example of a constrained $\Omega$

**Constraints** explicitely or implicitely define the feasible solution set

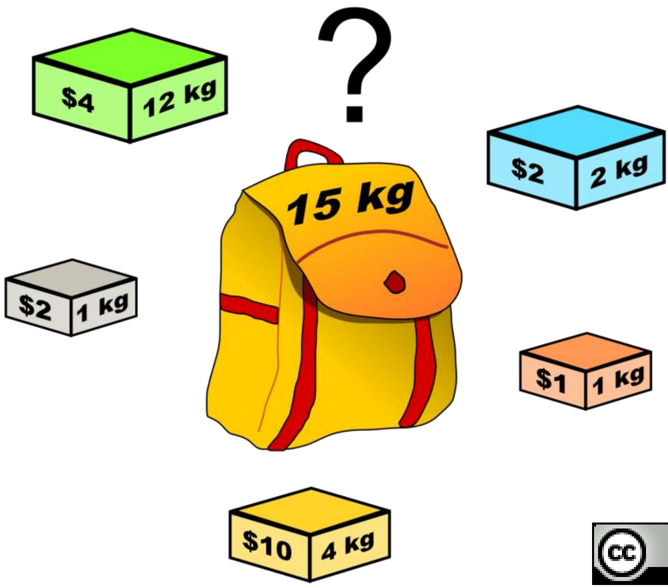
[e.g. ||x|| - 7 ≤ 0 vs. every solution should have at least 5 zero entries]

Hard constraints *must* be satisfied while soft constraints are preferred to hold but are not required to be satisfied

[e.g. constraints related to manufactoring precisions vs. cost constraints]

## Knapsack Problem

- Given a set of objects with a given weight and value (profit)
- Find a subset of objects whose overall mass is below a certain limit and maximizing the total value of the objects






*[Problem of ressource allocation with financial constraints]*

$$\max. \sum_{j=1}^{n} p_j x_j \text{ with } x_j \in \{0,1\}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \leq W$$

$$\Omega = \{0,1\}^n$$

# Example 2: Combinatorial Optimization

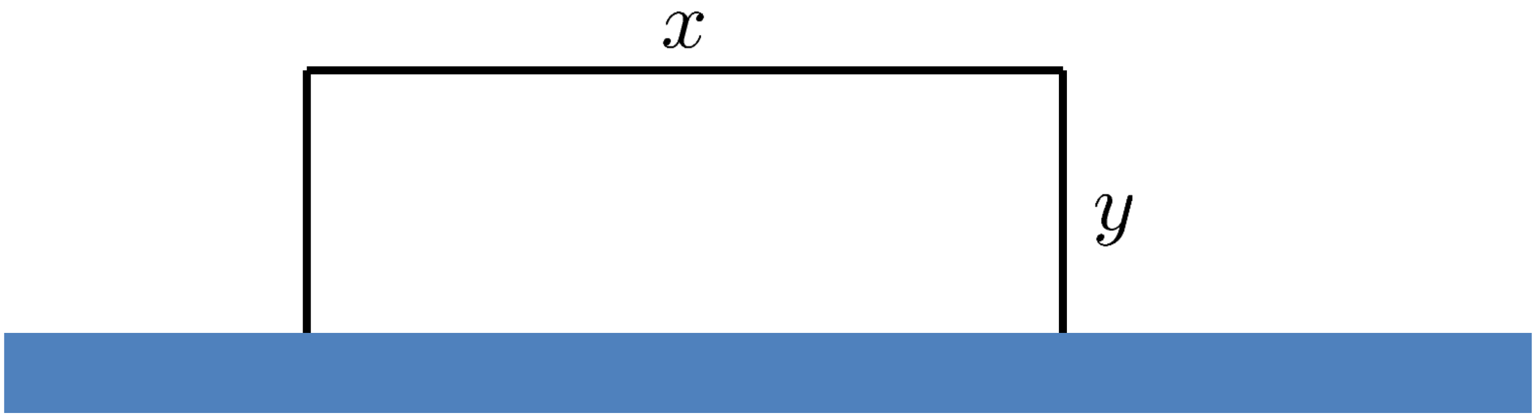
## Traveling Salesperson Problem (TSP)

- Given a set of cities and their distances
- Find the shortest path going through all cities


$$\Omega = S_n \text{ (set of all permutations)}$$

# Example 3: Continuous Optimization

A farmer has 500m of fence to fence off a rectangular field that is adjacent to a river. What is the maximal area he can fence off?



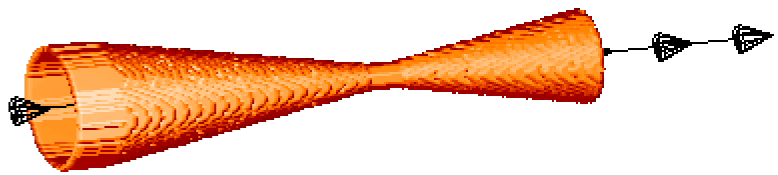

**Exercise:**

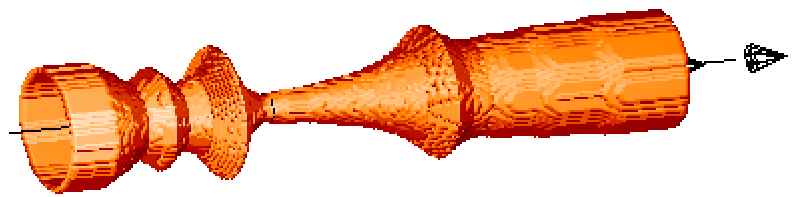
a) what is the search space?
b) what is the objective function?

# Example 4: A “Manual” Engineering Problem

## Optimizing a Two-Phase Nozzle [Schwefel 1968+]

- maximize thrust under constant starting conditions
- one of the first examples of Evolution Strategies

initial design: 

final design: 

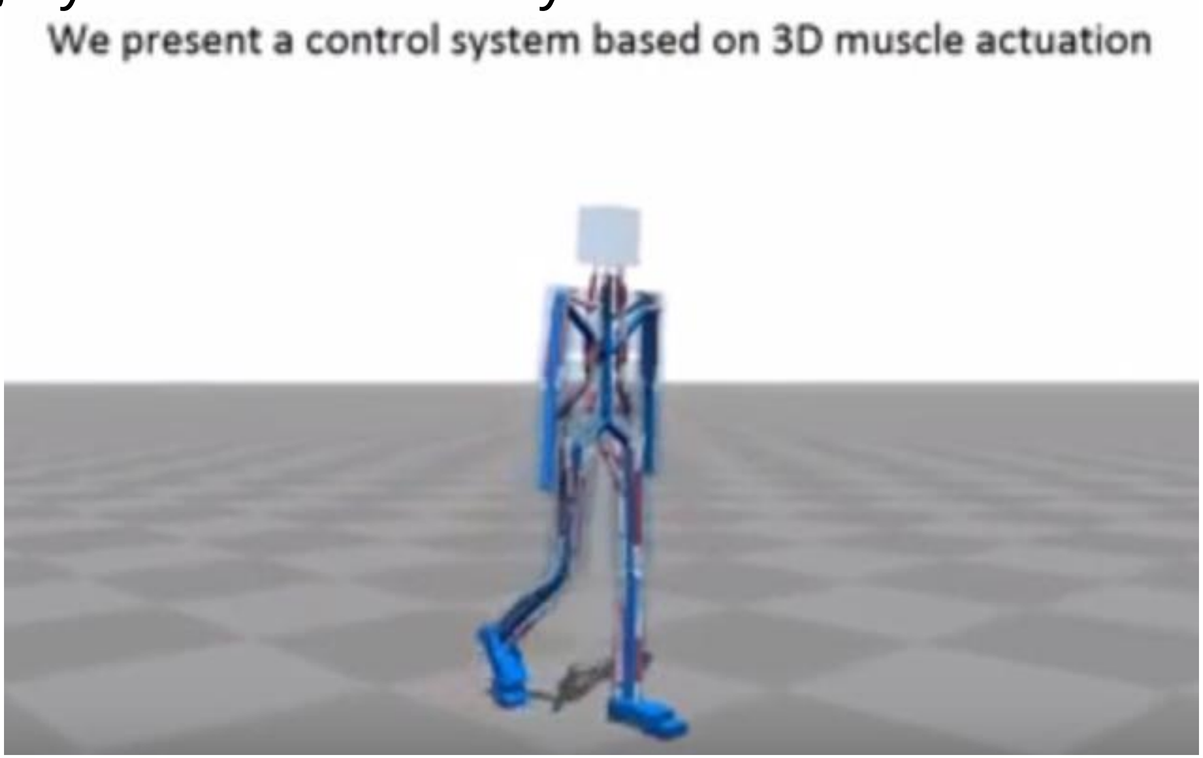
$\Omega =$ all possible nozzles of given number of slices

copyright Hans-Paul Schwefel

[http://ls11-www.cs.uni-dortmund.de/people/schwefel/EADemos/]

# Example 5: Continuous Optimization Problem

Computer simulation teaches itself to walk upright (virtual robots (of different shapes) learning to walk, through stochastic optimization (CMA-ES)), by Utrecht University:




https://www.youtube.com/watch?v=yci5FuI1ovk

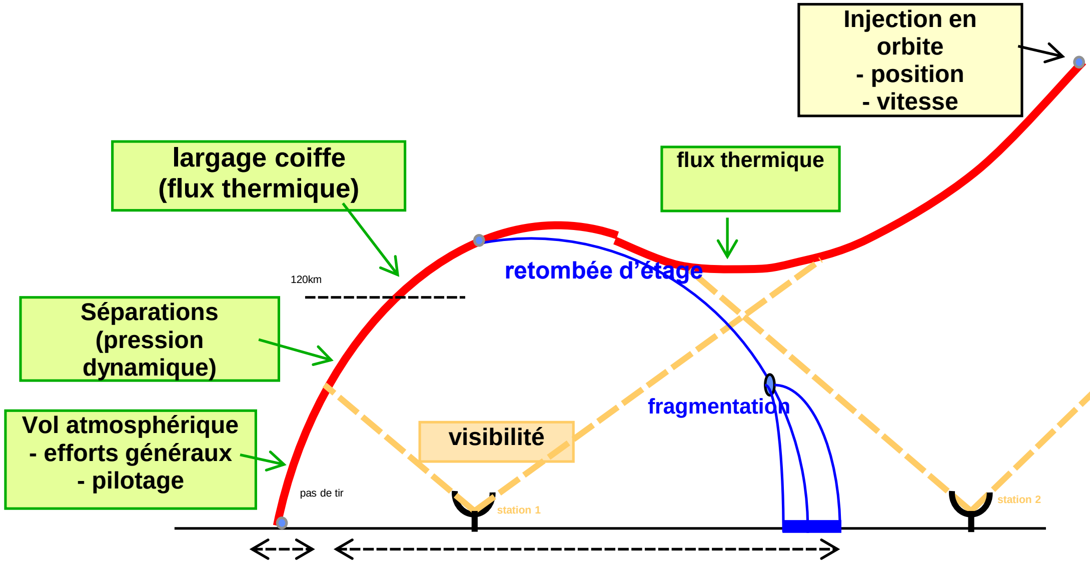
T. Geitjtenbeek, M. Van de Panne, F. Van der Stappen: "Flexible Muscle-Based Locomotion for Bipedal Creatures", SIGGRAPH Asia, 2013.

# Example 6: Constrained Continuous Optimization

## Design of a Launcher



Poppy




copyright by Astrium

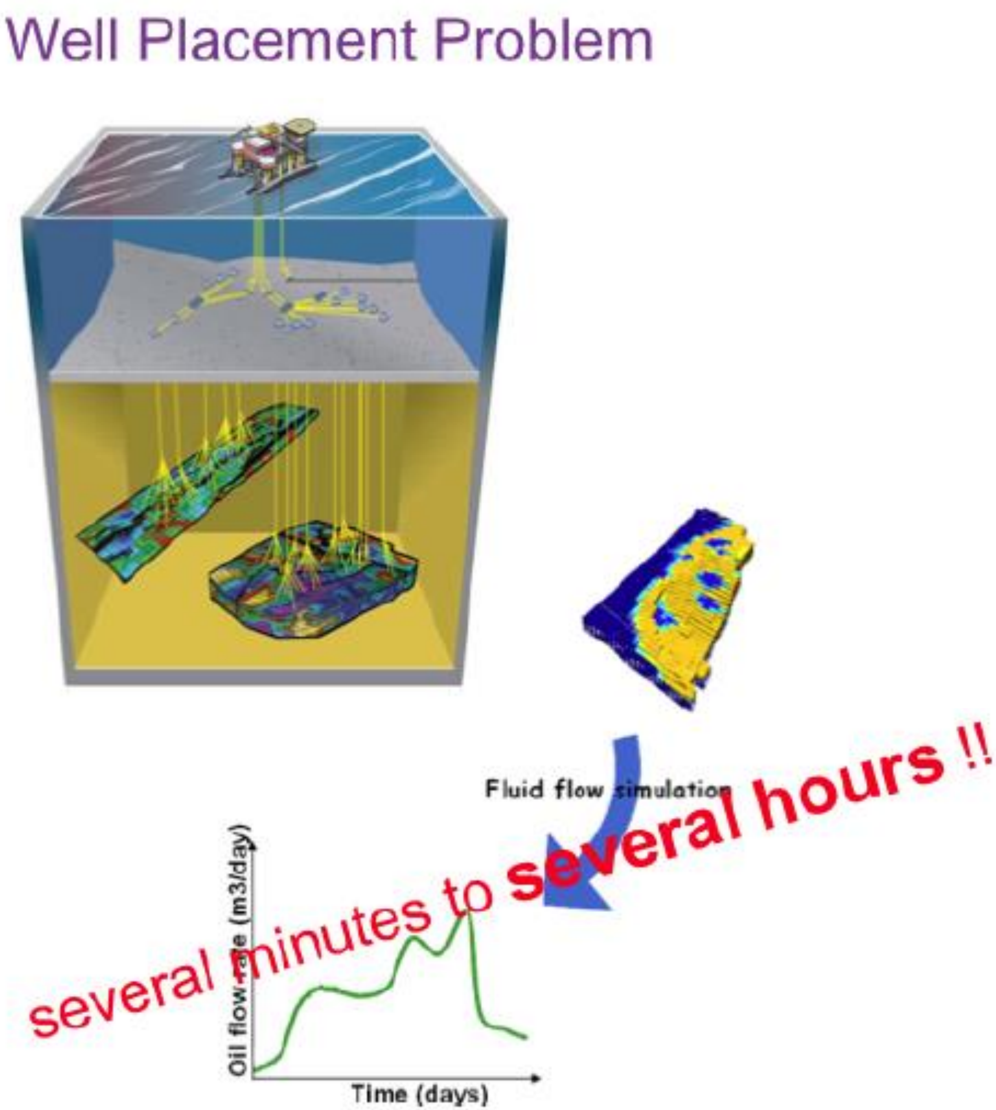
$$\Omega = \mathbb{R}^{23}$$

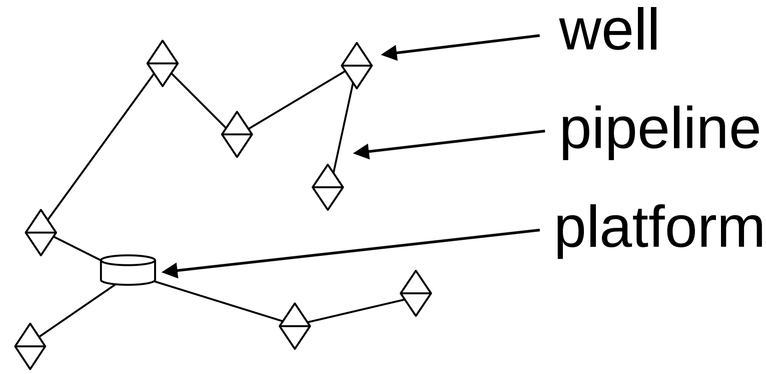
- Scenario: multi-stage launcher brings a satellite into orbit
- Minimize the overall cost of a launch
- Parameters: propellant mass of each stage / diameter of each stage / flux of each engine / parameters of the command law

*23 continuous parameters to optimize*

*+ constraints*

# Example 7: An Expensive Real-World Problem




for a given structure,
per well:

- angle & distance to previous well
- well depth

structure + $\mathbb{R}^{3 \cdot \#\text{wells}}$

$\Omega$: variable length!

# Example 8: Data Fitting – Data Calibration

## Objective

- Given a sequence of data points $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}, i = 1, \dots, N$, find a model "$y = f(\boldsymbol{x})$" that explains the data

*experimental measurements in biology, chemistry, ...*

- In general, choice of a parametric model or family of functions $(f_\theta)_{\theta \in \mathbb{R}^n}$

*use of expertise for choosing model*

*or only a simple model is affordable (e.g. linear, quadratic)*

- Try to find the parameter $\theta \in \mathbb{R}^n$ fitting best to the data

## Fitting best to the data

Minimize the quadratic error:

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^{N} |f_\theta(\boldsymbol{x}_i) - y_i|^2$$

# Example 9: Lin. Regression in Machine Learning

**Supervised Learning:**

Predict $y \in \mathcal{Y}$ from $\boldsymbol{x} \in \mathcal{X}$, given a set of observations (examples) $\{y_i, \boldsymbol{x}_i\}_{i=1,\dots,N}$

**(Simple) Linear regression where all the $y_i$ and $x_i$ are from $\mathbb{R}$**

Given a set of data: $\{y_i, \underbrace{x_i^1, \dots, x_i^p}_{\boldsymbol{x}_i^T}\}_{i=1\dots N}$

$$\min_{\boldsymbol{w}\in\mathbb{R}^p, \beta\in\mathbb{R}} \underbrace{\sum_{i=1}^{N} |\, \boldsymbol{w}^T \boldsymbol{x}_i + \beta - y_i \,|^2}_{||\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{w}} - \mathbf{y}||^2}$$

$\widetilde{\boldsymbol{X}} \in \mathbb{R}^{N\times(p+1)}$, $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{p+1}$

same as data fitting with linear model, i.e. $f_{(\boldsymbol{w},\beta)}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + \beta$,

$\theta \in \mathbb{R}^{p+1}$

# Example 9b: Regression

## General Regression for Arbitrary Data

- *Data:* $N$ observations $\{y_i, x_i\} \in \mathbb{R} \times \mathcal{X}$
- $\mathcal{X}$ not necessarily in $\mathbb{R}^n$
- $\Phi(x_i) \in \mathbb{R}^p$ features of $x_i$
- prediction as a linear function of the feature $\hat{y} = \langle \theta, \Phi(x) \rangle$
- *empirical risk minimization:* find $\hat{\theta}$ solution of

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} I(y_i, \langle \theta, \Phi(x_i) \rangle)$$

where $I$ is a loss function [example: quadratic loss $I(y, \hat{y}) = 1/2(y - \hat{y})^2$ ]

# Example 10: Interactive Optimization

## Coffee Tasting Problem

- Find a mixture of coffee in order to keep the coffee taste from one year to another
- Objective function = opinion of one expert






*M. Herdy: "Evolution Strategies with subjective selection", 1996*

# Many Problems, Many Algorithms?

## Observation:

- Many problems with different properties
- For each, it seems a different algorithm?

## In Practice:

- often most important to categorize your problem first in order to find / develop the right method
- → problem types

Algorithm design is an art,
what is needed is skill, intuition, luck, experience,
special knowledge and craft

freely translated and adapted from Ingo Wegener (1950-2008)

# Problem Types

- discrete vs. continuous
  - discrete: integer (linear) programming vs. combinatorial problems
  - continuous: linear, quadratic, smooth/nonsmooth, blackbox/DFO, ...
  - both discrete&continuous variables: mixed integer problem
- constrained vs. unconstrained
- one or multiple objective functions

Not covered in this introductory lecture:

- deterministic vs. stochastic outcome of objective function(s)

# Numerical Blackbox Optimization

Typical scenario in the continuous case:

Optimize $f: \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}^k$


*derivatives not available or not useful*

# General Concepts in Optimization

- search domain
  - discrete vs. continuous variables vs. mixed integer
  - finite vs. infinite dimension
- constraints
  - bounds
  - linear/quadratic/non-linear constraint
  - blackbox constraint

Further important aspects (in practice):

- deterministic vs. stochastic algorithms
- exact vs. approximation algorithms vs. heuristics
- anytime algorithms
- simulation-based optimization problem / expensive problem

# continuous optimization

# Continuous Optimization

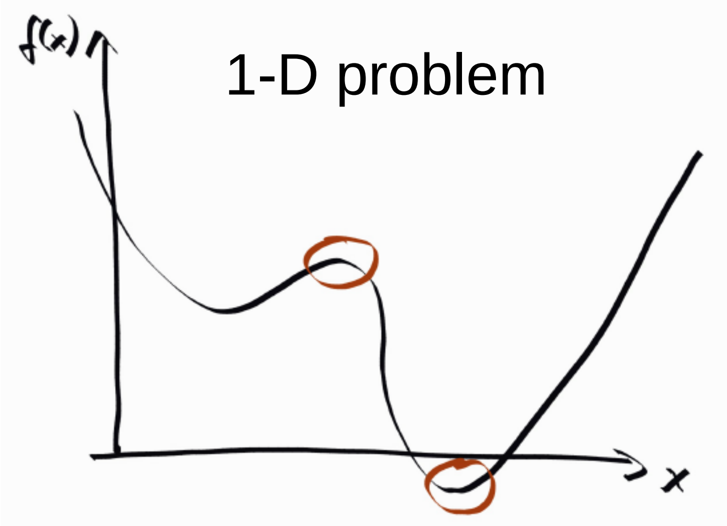
- Optimize $f$: $\begin{cases} \Omega \subset \mathbb{R}^n \to \mathbb{R} \\ x = (x_1, \dots, x_n) \to f(x_1, \dots, x_n) \end{cases}$

  $\in \mathbb{R}$

  *unconstrained* optimization

- Search space is continuous, i.e. composed of real vectors $x \in \mathbb{R}^n$

- $n = \begin{cases} \text{dimension of the problem} \\ \text{dimension of the search space } \mathbb{R}^n \text{ (as vector space)} \end{cases}$




2-D level sets

# Unconstrained vs. Constrained Optimization

## Unconstrained optimization

$$\inf \{f(x) \mid x \in \mathbb{R}^n\}$$

## Constrained optimization

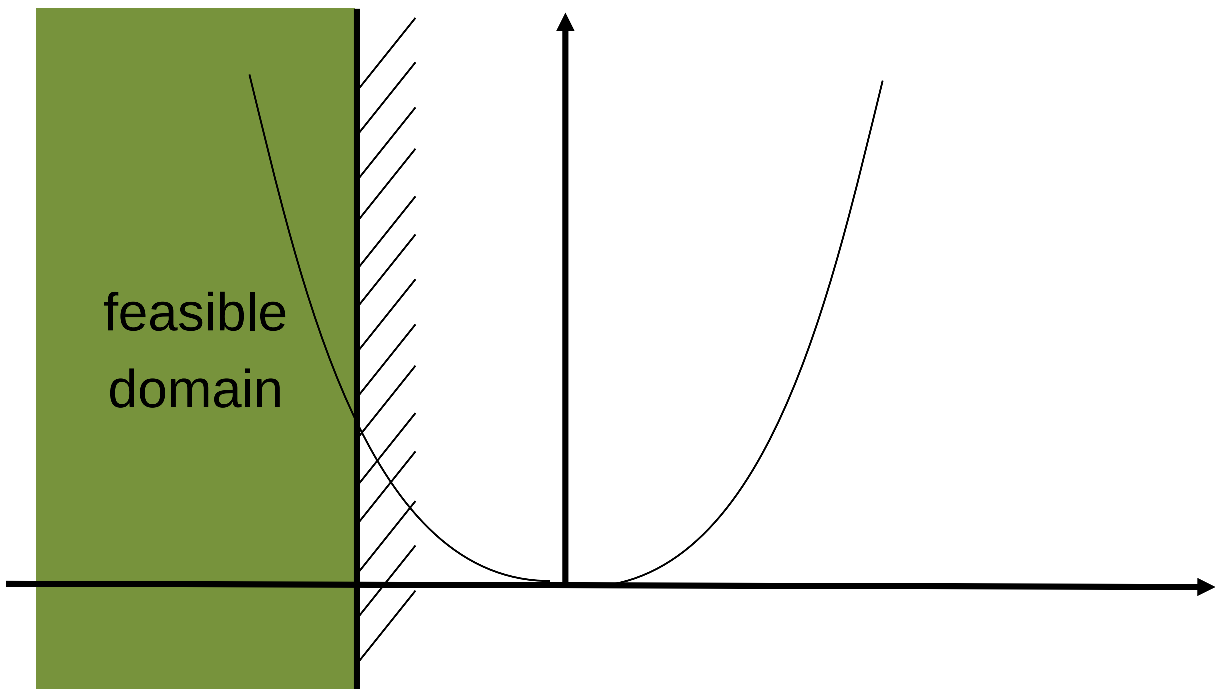
- Equality constraints: $\inf \{f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0, 1 \leq k \leq p\}$

- Inequality constraints: $\inf \{f(x) \mid x \in \mathbb{R}^n, g_k(x) \leq 0, 1 \leq k \leq p\}$

where always $g_k \colon \mathbb{R}^n \to \mathbb{R}$

# Example of a Constraint

$$\min_{x \in \mathbb{R}} f(x) = x^2 \text{ such that } x \leq -1$$



# Analytical Functions

**Example: 1-D**

$$f_1(x) = a(x - x_0)^2 + b$$

where $x, x_0, b \in \mathbb{R}, a \in \mathbb{R}$

**Generalization:**

convex quadratic function

$$f_2(x) = (x - x_0)^T A \, (x - x_0) + b$$

where $x, x_0, b \in \mathbb{R}^n, A \in \mathbb{R}^{\{\mathrm{n} \times n\}}$

and $A$ symmetric positive definite (SPD)

**Exercise:**
What is the minimum of $f_2(x)$?

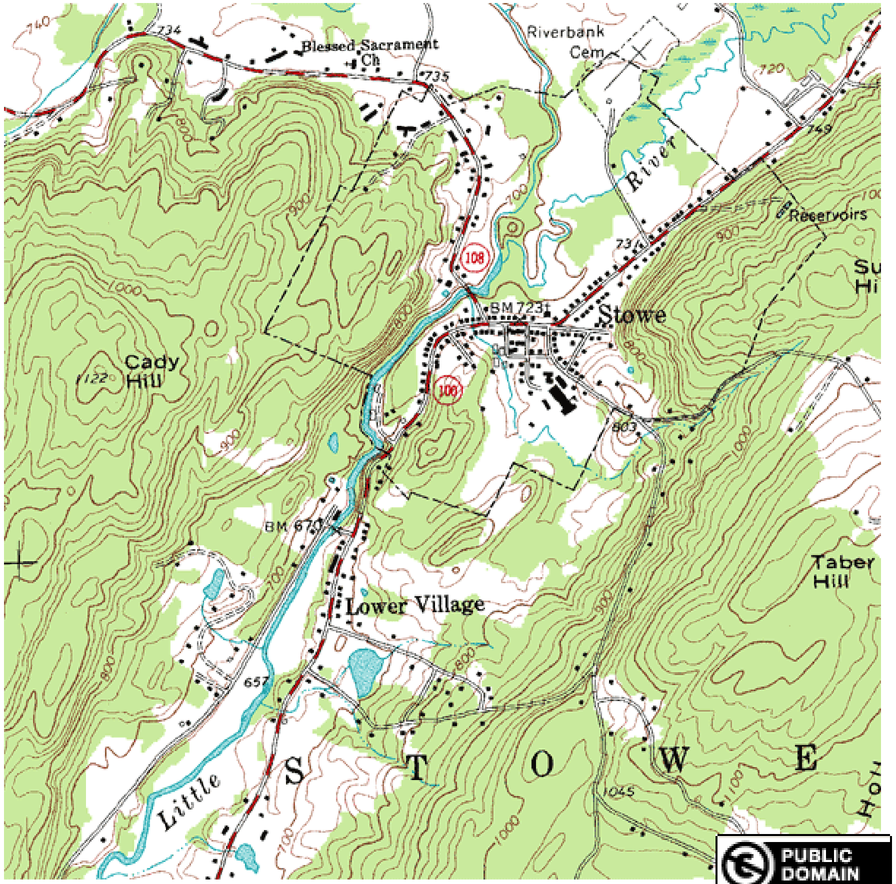
# Levels Sets of Convex Quadratic Functions

**Continuation of exercise:**
What are the level sets of $f_2$?

**Reminder:** level sets of a function

$$L_c = \{x \in \mathbb{R}^n \mid f(x) = c\}$$

(similar to topography lines / level sets on a map)



# Levels Sets of Convex Quadratic Functions

**Continuation of exercise:**
What are the level sets of $f_2$?

- Probably too complicated in general, thus an example here
- Consider $A = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix}, b = 0, n = 2$
  a) Compute $f_2(x)$.
  b) Plot the level sets of $f_2(x)$.
  c) More generally, for $n = 2$, if $A$ is SPD with eigenvalues $\lambda_1 = 9$ and $\lambda_2 = 1$, what are the level sets of $f_2(x)$?

# What Makes a Function Difficult to Solve?

- dimensionality

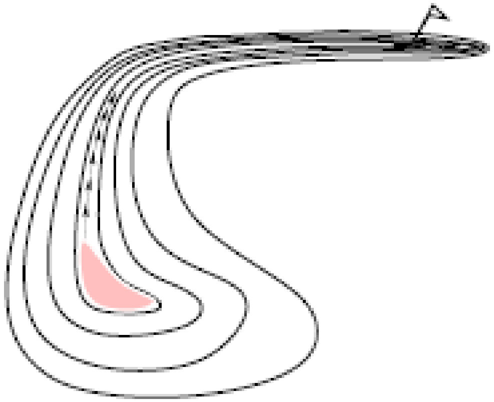
  *(considerably) larger than three*

- non-separability

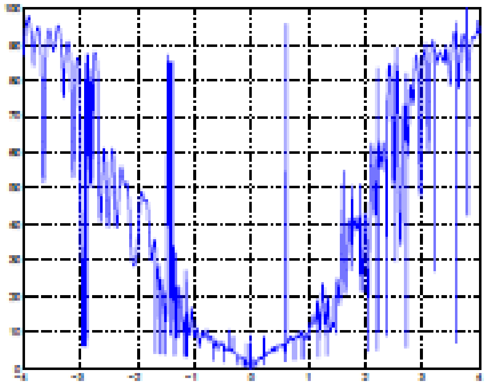
  *dependencies between the objective variables*

- ill-conditioning
- ruggedness

  *non-smooth, discontinuous, multimodal, and/or noisy function*



a narrow ridge



cut from 3D example, solvable with an evolution strategy

# Curse of Dimensionality

- The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

- Example: Consider placing 100 points onto a real interval, say $[0,1]$. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[0,1]^{10}$ would require $100^{10} = 10^{20}$ points. The original 100 points appear now as isolated points in a vast empty space.

- Consequently, a search policy (e.g. exhaustive search) that is valuable in small dimensions might be useless in moderate or large dimensional search spaces.

# Separable Problems

## Definition (Separable Problem)

A function $f$ is separable if

$$\underset{(x_1,\ldots,x_n)}{\operatorname{argmin}} f(x_1,\ldots,x_n) = \left(\underset{x_1}{\operatorname{argmin}} f(x_1,\ldots),\ldots,\underset{x_n}{\operatorname{argmin}} f(\ldots,x_n)\right)$$

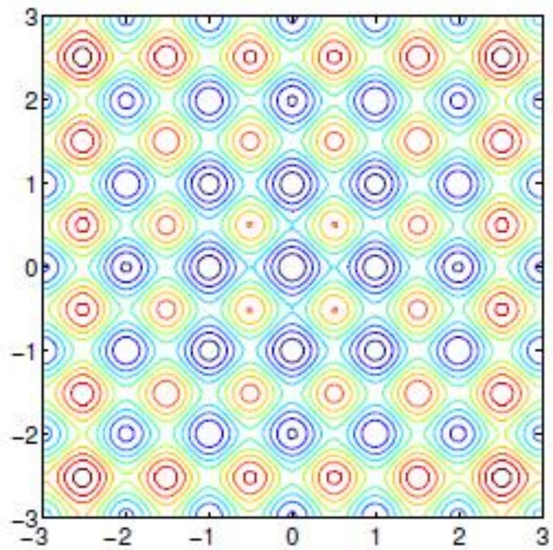
*$\Longrightarrow$ it follows that $f$ can be optimized in a sequence of $n$ independent 1-D optimization processes*

## Example:

Additively decomposable functions

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i)$$

*Rastrigin function*



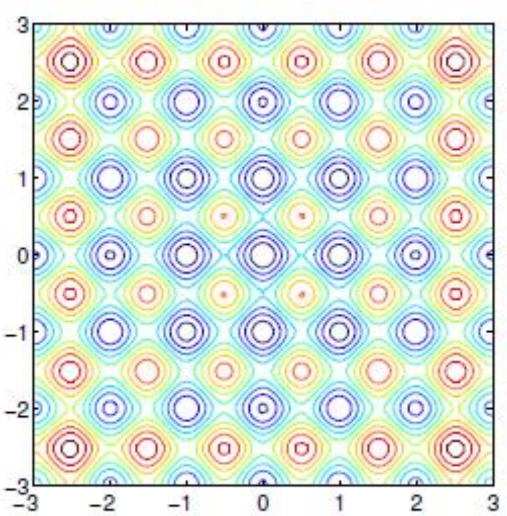
# Non-Separable Problems

Building a non-separable problem from a separable one [1,2]

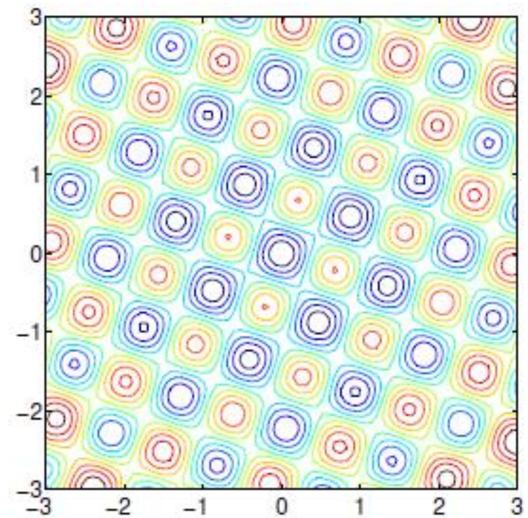
## Rotating the coordinate system

- $f: \boldsymbol{x} \mapsto f(\boldsymbol{x})$ separable
- $f: \boldsymbol{x} \mapsto f(R\boldsymbol{x})$ non-separable

$R$ rotation matrix




$$\xrightarrow{R}$$



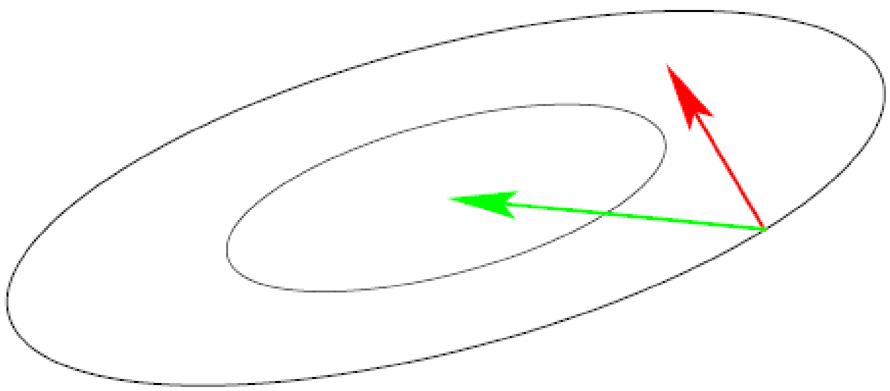

[1] N. Hansen, A. Ostermeier, A. Gawelczyk (1995). "On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation". Sixth ICGA, pp. 57-64, Morgan Kaufmann
[2] R. Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

# Ill-Conditioned Problems: Curvature of Level Sets

Consider the convex-quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^T H(\boldsymbol{x}-\boldsymbol{x}^*) = \frac{1}{2}\sum_i h_{i,i}x_i^2 + \frac{1}{2}\sum_{i,j} h_{i,j}x_i x_j$$

H is Hessian matrix of $f$ and symmetric positive definite



gradient direction $-f'(x)^T$

Newton direction $-H^{-1}f'(x)^T$

*Ill-conditioning means squeezed level sets (high curvature). Condition number equals nine here. Condition numbers up to $10^{10}$ are not unusual in real-world problems.*

If $H \approx I$ (small condition number of $H$) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of $H^{-1}$) information necessary.

# Different Notions of Optimum

## Unconstrained case

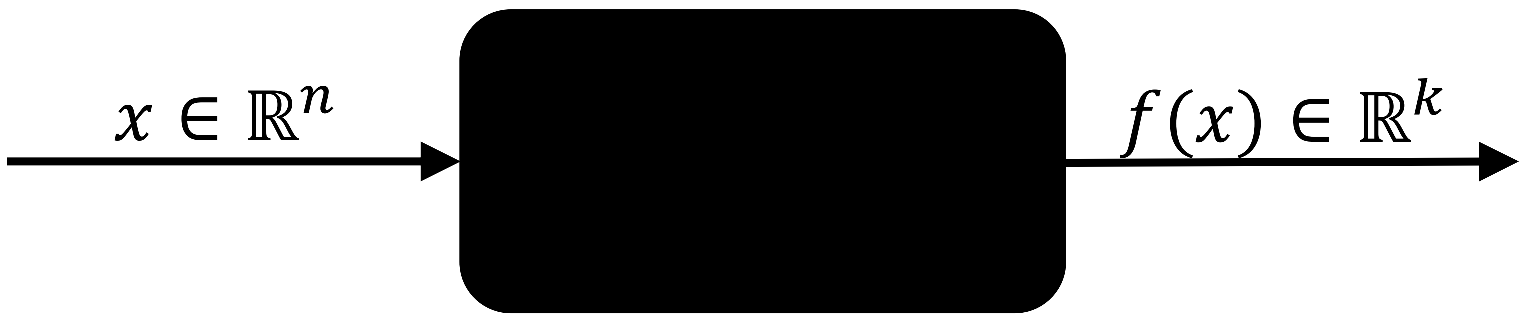
- local vs. global
  - local minimum $\boldsymbol{x}^*$: $\exists$ a neighborhood $V$ of $\boldsymbol{x}^*$ such that
    $\forall \boldsymbol{x} \in \mathrm{V}: f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$
  - global minimum: $\forall \boldsymbol{x} \in \Omega: f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$
- strict local minimum if the inequality is strict

# Blackbox optimization benchmarking

...and some more details on the group project

# Numerical Blackbox Optimization

Optimize $f: \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}^k$




*derivatives not available or not useful*

**Not clear:**

which of the many algorithms should I use on my problem?

# Numerical Blackbox Optimizers

## Deterministic algorithms

Quasi-Newton with estimation of gradient (**BFGS**) [Broyden et al. 1970]
Simplex downhill [Nelder & Mead 1965]
Pattern search [Hooke and Jeeves 1961]
Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]

## Stochastic (randomized) search methods

Evolutionary Algorithms (continuous domain)

- Differential Evolution [Storn & Price 1997]
- Particle Swarm Optimization [Kennedy & Eberhart 1995]
- **Evolution Strategies, CMA-ES** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Estimation of Distribution Algorithms (EDAs) [Larrañaga, Lozano, 2002]
- Cross Entropy Method (same as EDA) [Rubinstein, Kroese, 2004]
- Genetic Algorithms [Holland 1975, Goldberg 1989]

Simulated annealing [Kirkpatrick et al. 1983]
Simultaneous perturbation stochastic approx. (SPSA) [Spall 2000]

# Numerical Blackbox Optimizers

## Deterministic algorithms

Quasi-Newton with estimation of gradient (**BFGS**) [Broyden et al. 1970]
Simplex downhill [Nelder & Mead 1965]
Pattern search [Hooke and Jeeves 1961]
Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]

choice typically not immediately clear although practitioners have knowledge about which difficulties their problem has (e.g. multi-modality, non-separability, ...)

- **Evolution Strategies, CMA-ES** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Estimation of Distribution Algorithms (EDAs) [Larrañaga, Lozano, 2002]
- Cross Entropy Method (same as EDA) [Rubinstein, Kroese, 2004]
- Genetic Algorithms [Holland 1975, Goldberg 1989]

Simulated annealing [Kirkpatrick et al. 1983]
Simultaneous perturbation stochastic approx. (SPSA) [Spall 2000]

# Need: Benchmarking

- understanding of algorithms
- algorithm selection
- putting algorithms to a standardized test
  - simplify judgement
  - simplify comparison
  - regression test under algorithm changes

Kind of everybody has to do it (and it is tedious):

- choosing (and implementing) problems, performance measures, visualization, stat. tests, ...
- running a set of algorithms

# that's where COCO comes into play

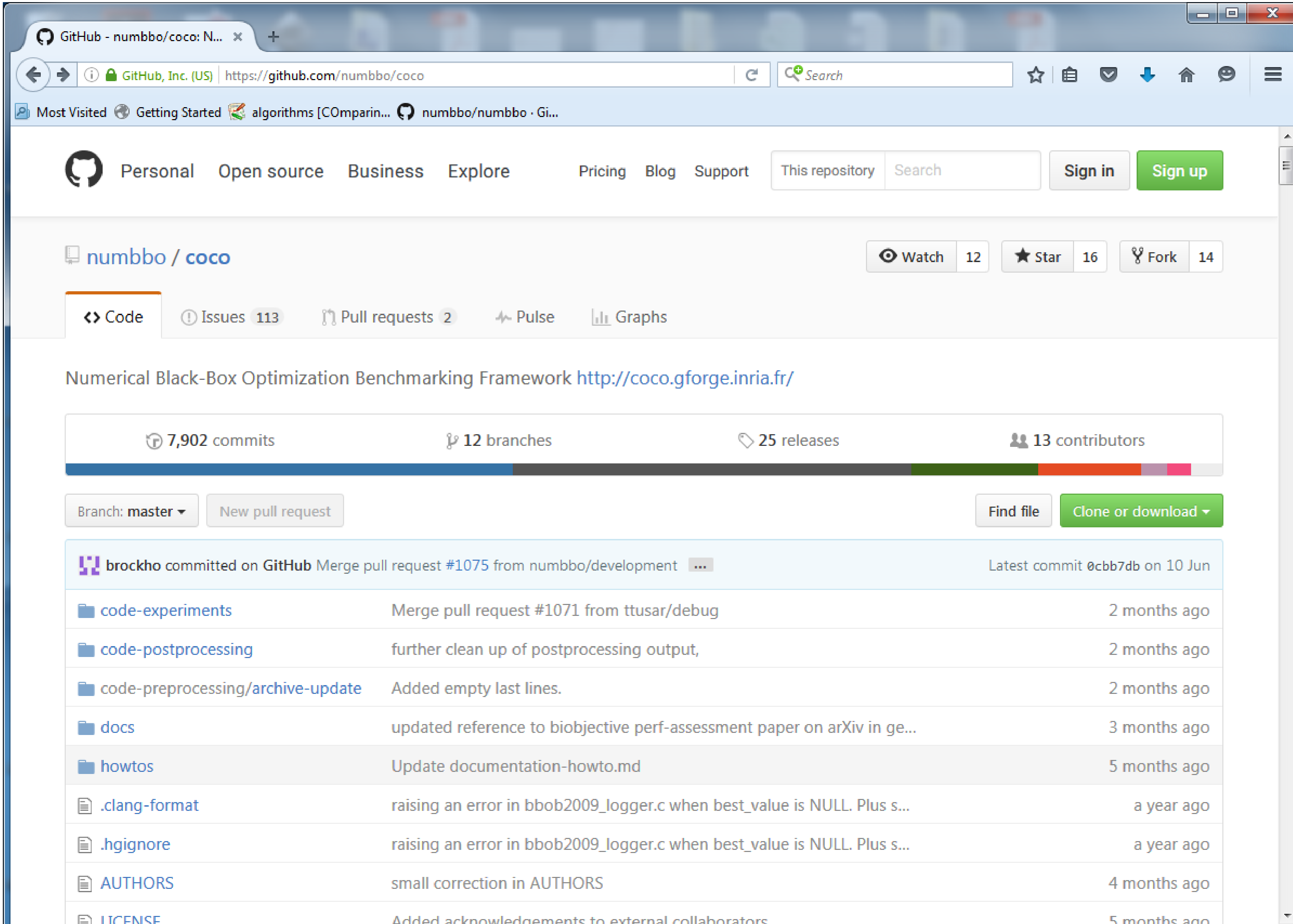
**Comparing Continuous Optimizers Platform**

**`https://github.com/numbbo/coco`**

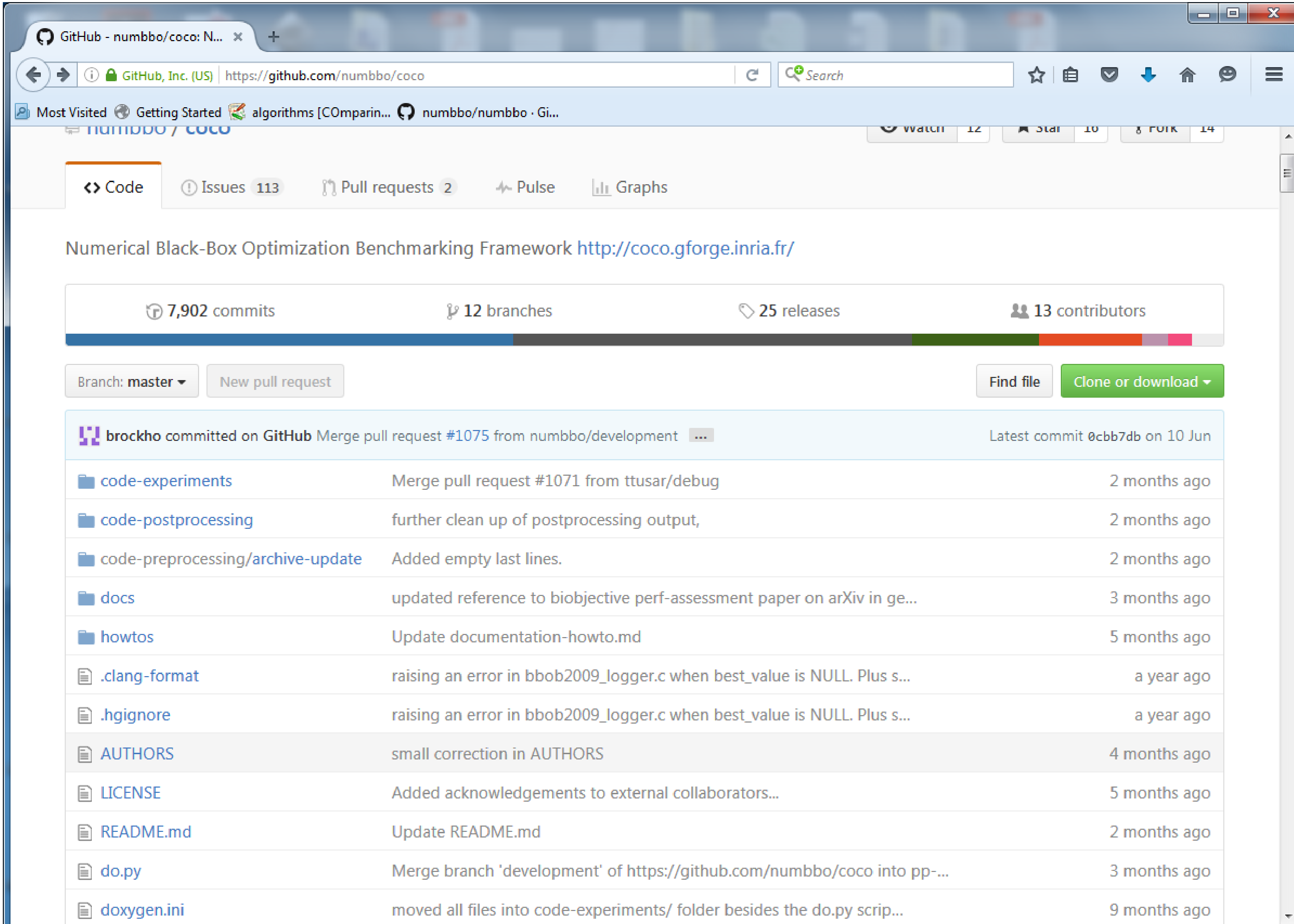
# automatized benchmarking

# How to benchmark algorithms with COCO?

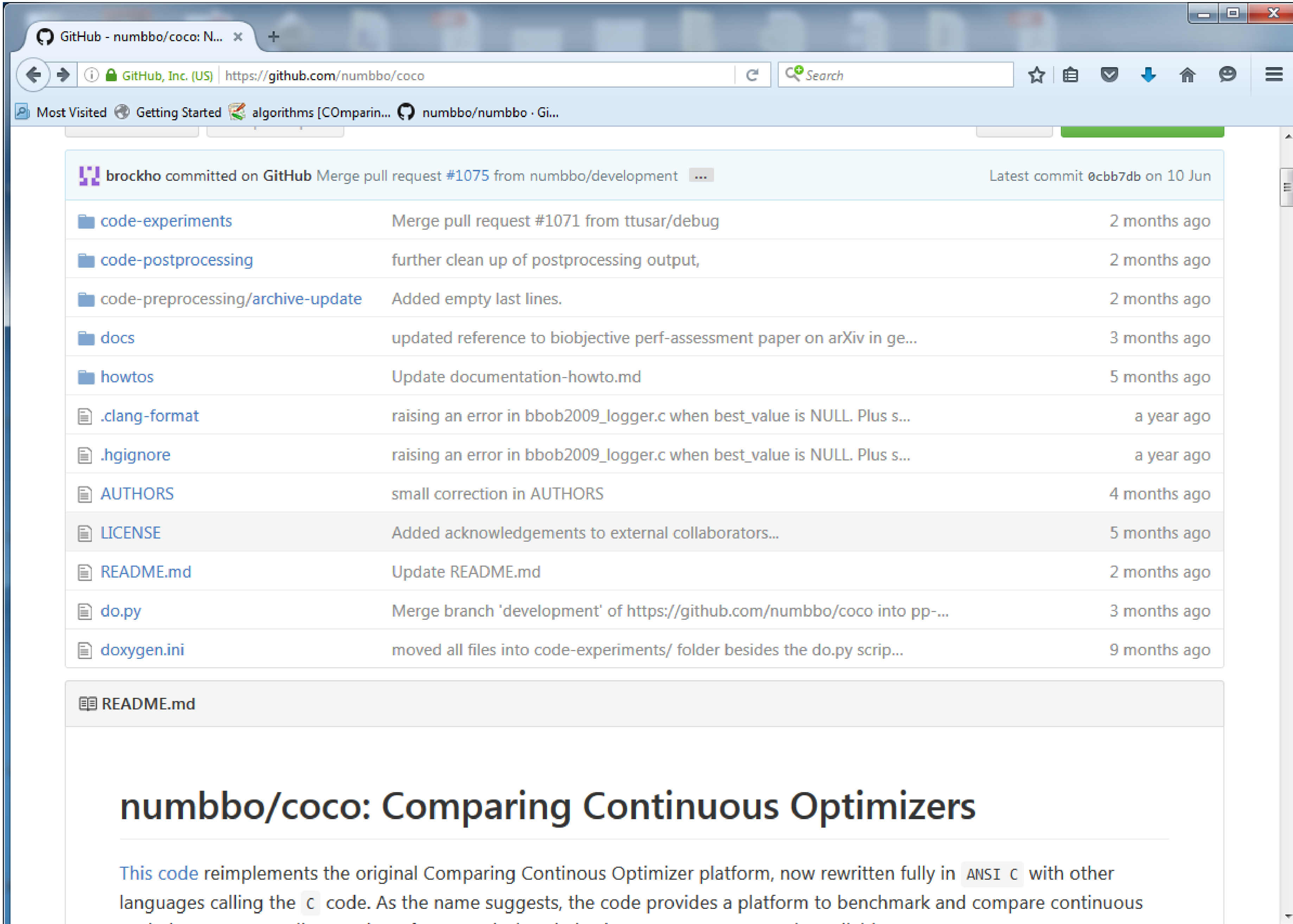
# https://github.com/numbbo/coco



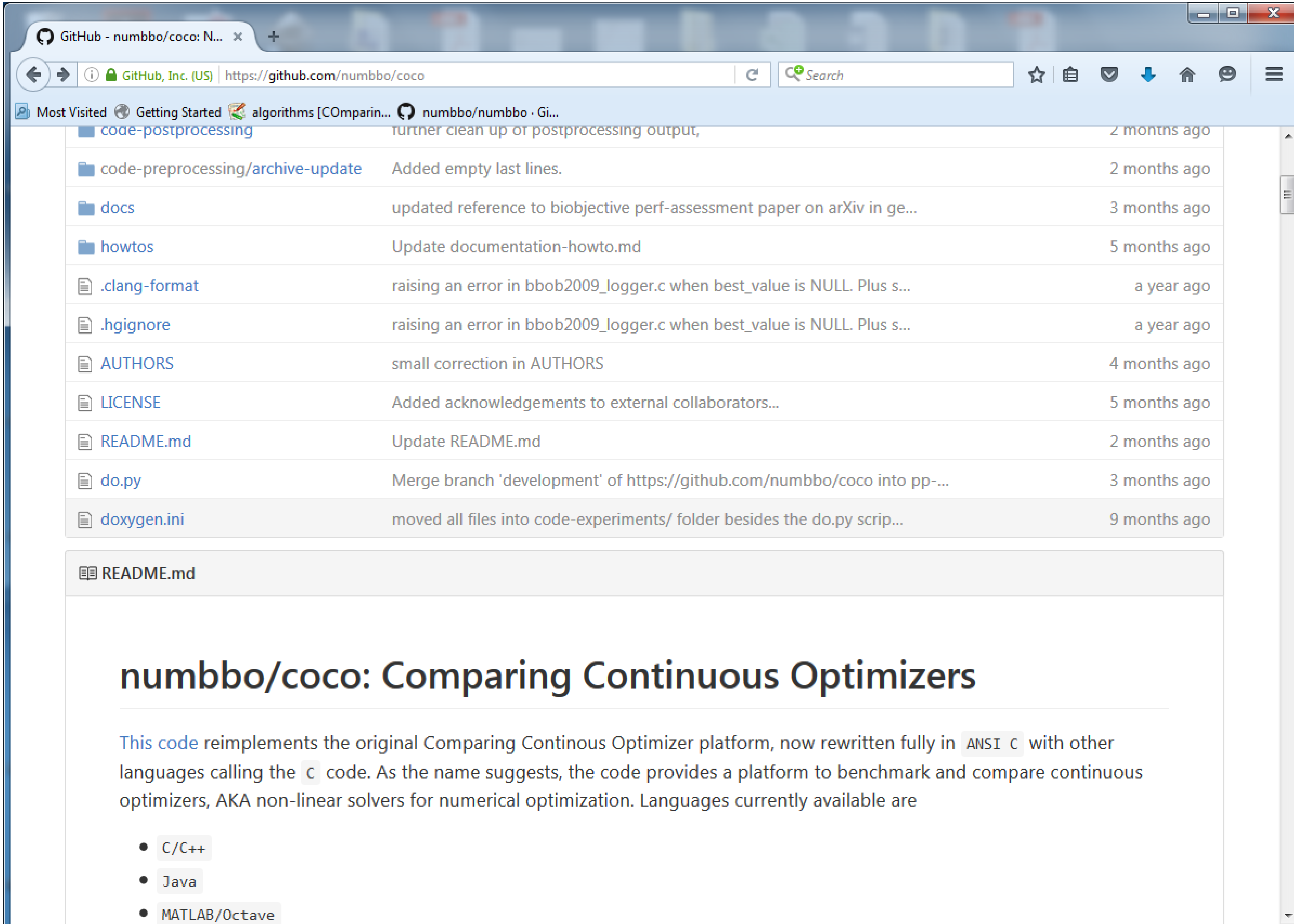
# https://github.com/numbbo/coco



# https://github.com/numbbo/coco



https://github.com/numbbo/coco



GitHub - numbbo/coco: N...

GitHub, Inc. (US) https://github.com/numbbo/coco

Most Visited | Getting Started | algorithms [COmparin... | numbbo/numbbo · Gi...

| | | |
|---|---|---|
| code-postprocessing | further clean up of postprocessing output, | 2 months ago |
| code-preprocessing/archive-update | Added empty last lines. | 2 months ago |
| docs | updated reference to biobjective perf-assessment paper on arXiv in ge... | 3 months ago |
| howtos | Update documentation-howto.md | 5 months ago |
| .clang-format | raising an error in bbob2009_logger.c when best_value is NULL. Plus s... | a year ago |
| .hgignore | raising an error in bbob2009_logger.c when best_value is NULL. Plus s... | a year ago |
| AUTHORS | small correction in AUTHORS | 4 months ago |
| LICENSE | Added acknowledgements to external collaborators... | 5 months ago |
| README.md | Update README.md | 2 months ago |
| do.py | Merge branch 'development' of https://github.com/numbbo/coco into pp-... | 3 months ago |
| doxygen.ini | moved all files into code-experiments/ folder besides the do.py scrip... | 9 months ago |

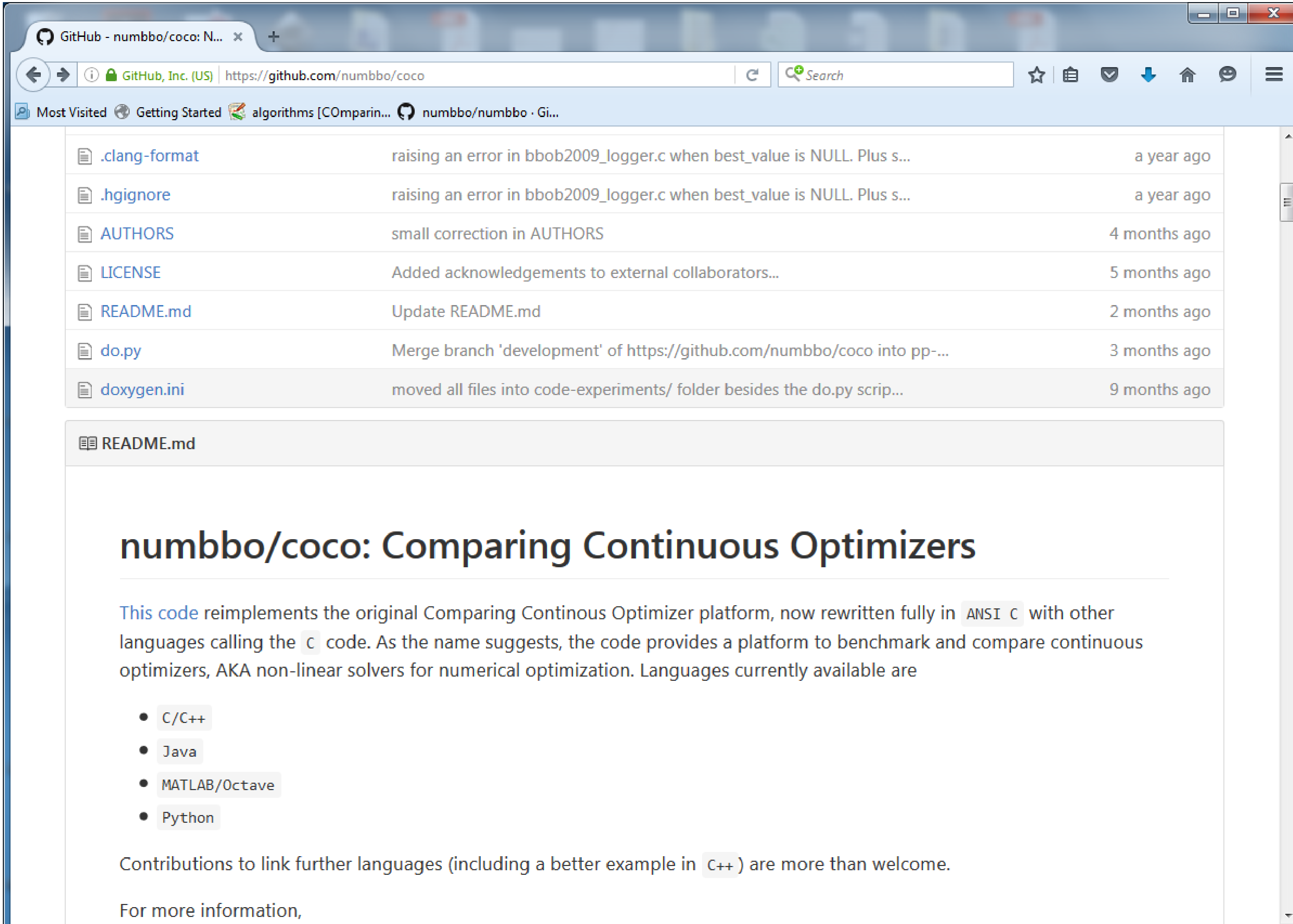
README.md

# numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

- `C/C++`
- `Java`
- `MATLAB/Octave`

# https://github.com/numbbo/coco



| | | |
|---|---|---|
| .clang-format | raising an error in bbob2009_logger.c when best_value is NULL. Plus s... | a year ago |
| .hgignore | raising an error in bbob2009_logger.c when best_value is NULL. Plus s... | a year ago |
| AUTHORS | small correction in AUTHORS | 4 months ago |
| LICENSE | Added acknowledgements to external collaborators... | 5 months ago |
| README.md | Update README.md | 2 months ago |
| do.py | Merge branch 'development' of https://github.com/numbbo/coco into pp-... | 3 months ago |
| doxygen.ini | moved all files into code-experiments/ folder besides the do.py scrip... | 9 months ago |

README.md

# numbbo/coco: Comparing Continuous Optimizers

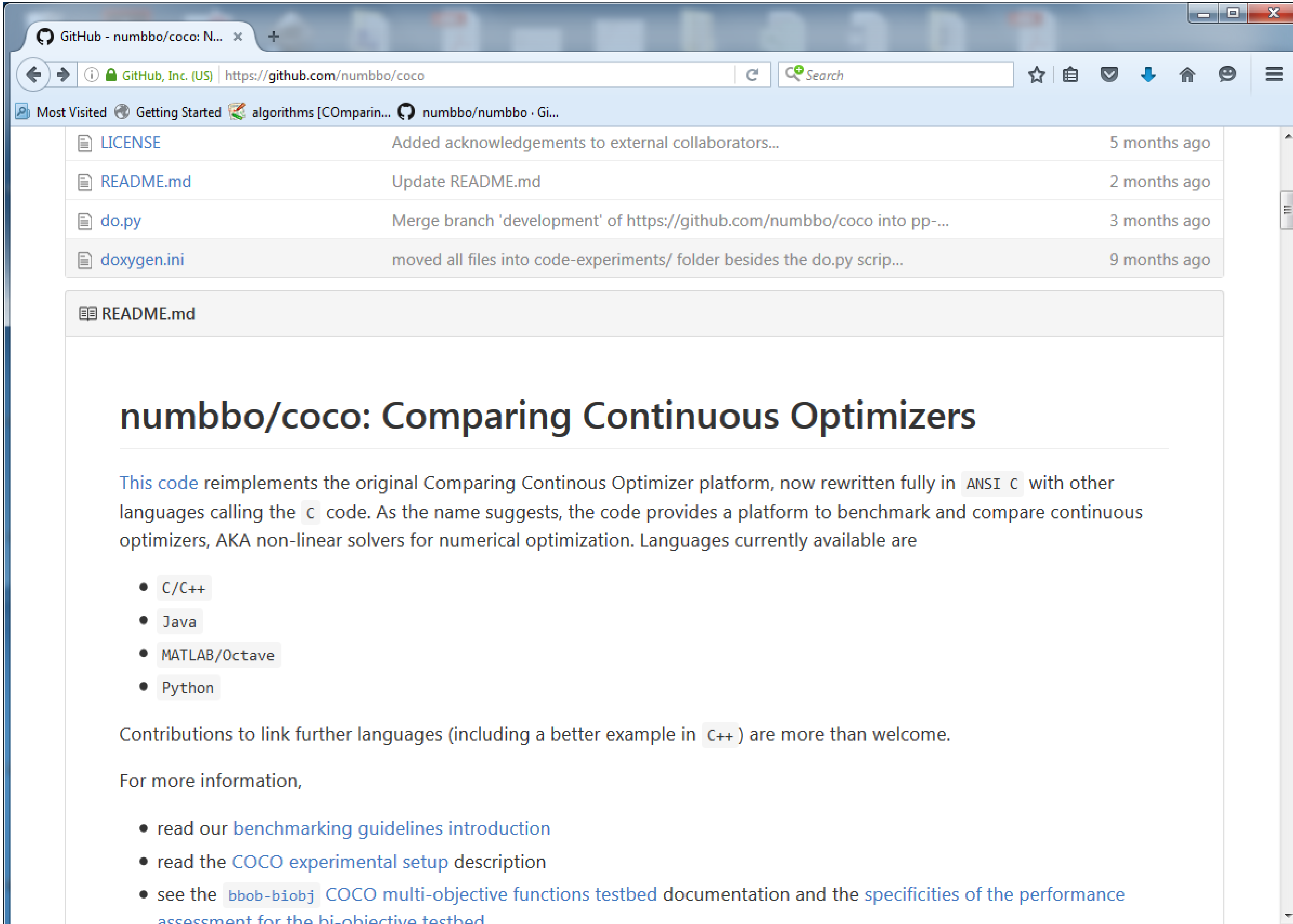
This code reimplements the original Comparing Continous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

Contributions to link further languages (including a better example in `C++`) are more than welcome.

For more information,

https://github.com/numbbo/coco




| | | |
|---|---|---|
| LICENSE | Added acknowledgements to external collaborators... | 5 months ago |
| README.md | Update README.md | 2 months ago |
| do.py | Merge branch 'development' of https://github.com/numbbo/coco into pp-... | 3 months ago |
| doxygen.ini | moved all files into code-experiments/ folder besides the do.py scrip... | 9 months ago |

README.md

# numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

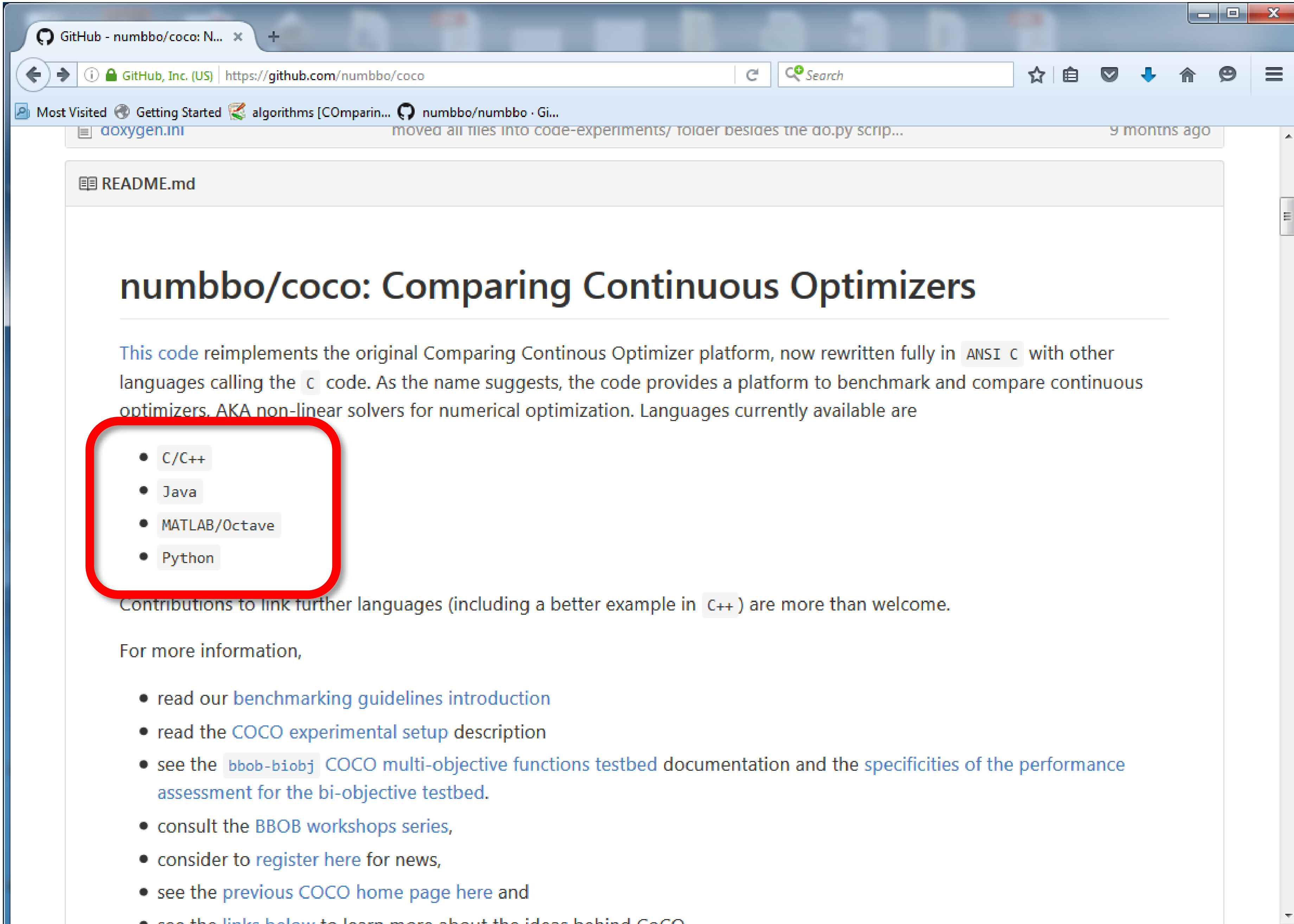
- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

Contributions to link further languages (including a better example in `C++`) are more than welcome.

For more information,

- read our benchmarking guidelines introduction
- read the COCO experimental setup description
- see the `bbob-biobj` COCO multi-objective functions testbed documentation and the specificities of the performance assessment for the bi-objective testbed

https://github.com/numbbo/coco



README.md

# numbbo/coco: Comparing Continuous Optimizers

This code reimplements the original Comparing Continous Optimizer platform, now rewritten fully in `ANSI C` with other languages calling the `C` code. As the name suggests, the code provides a platform to benchmark and compare continuous optimizers, AKA non-linear solvers for numerical optimization. Languages currently available are

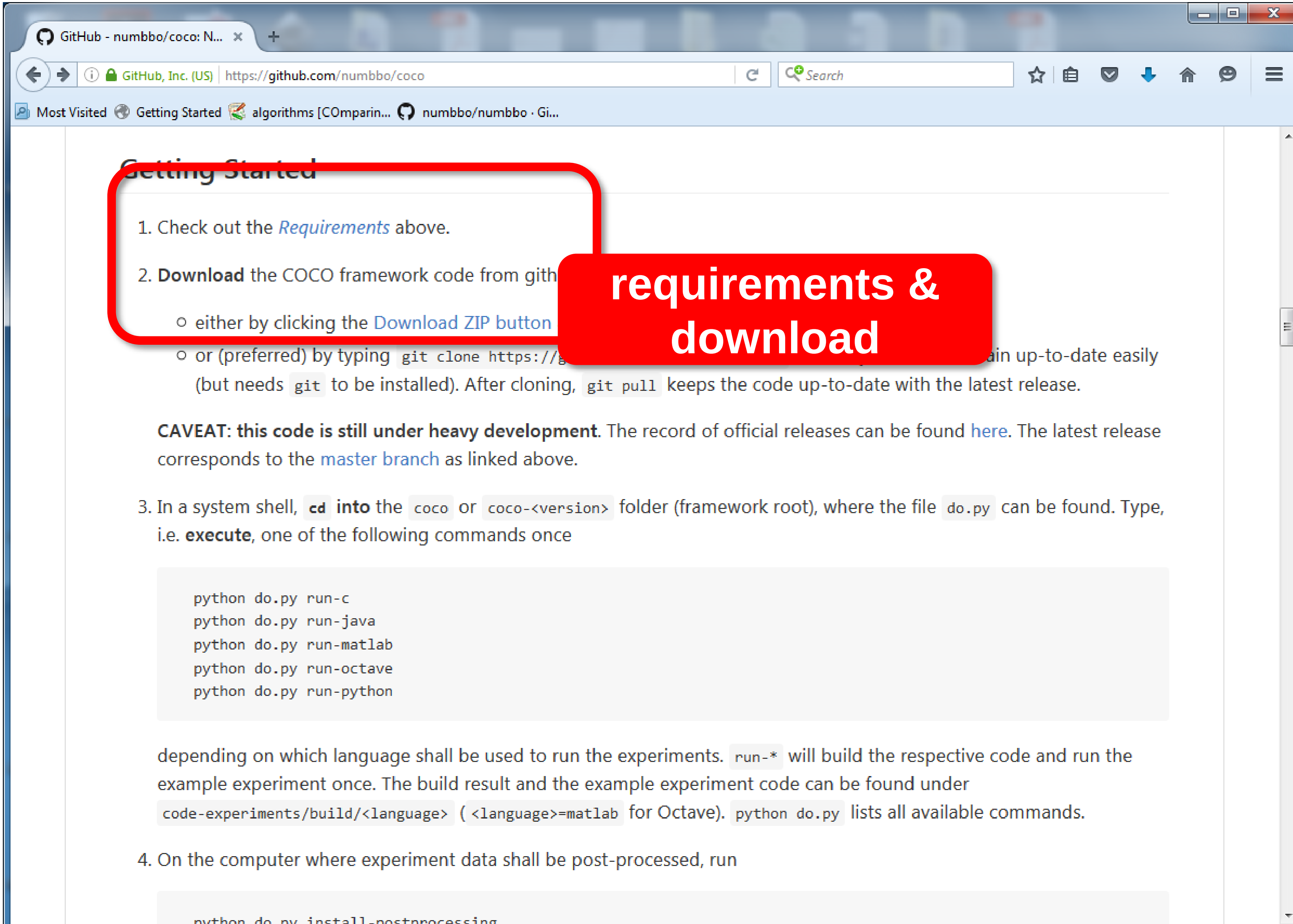
- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

Contributions to link further languages (including a better example in `C++`) are more than welcome.

For more information,

- read our benchmarking guidelines introduction
- read the COCO experimental setup description
- see the `bbob-biobj` COCO multi-objective functions testbed documentation and the specificities of the performance assessment for the bi-objective testbed.
- consult the BBOB workshops series,
- consider to register here for news,
- see the previous COCO home page here and

# https://github.com/numbbo/coco



GitHub - numbbo/coco: N...

https://github.com/numbbo/coco

## Getting Started

1. Check out the *Requirements* above.
2. **Download** the COCO framework code from gith
   - either by clicking the Download ZIP button
   - or (preferred) by typing `git clone https://g` ... ain up-to-date easily (but needs `git` to be installed). After cloning, `git pull` keeps the code up-to-date with the latest release.

   **CAVEAT: this code is still under heavy development**. The record of official releases can be found here. The latest release corresponds to the master branch as linked above.
3. In a system shell, **`cd` into** the `coco` or `coco-<version>` folder (framework root), where the file `do.py` can be found. Type, i.e. **execute**, one of the following commands once

```
python do.py run-c
python do.py run-java
python do.py run-matlab
python do.py run-octave
python do.py run-python
```

   depending on which language shall be used to run the experiments. `run-*` will build the respective code and run the example experiment once. The build result and the example experiment code can be found under `code-experiments/build/<language>` (`<language>=matlab` for Octave). `python do.py` lists all available commands.
4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

**requirements & download**

# https://github.com/numbbo/coco

GitHub - numbbo/coco: N...

GitHub, Inc. (US) | https://github.com/numbbo/coco

Most Visited | Getting Started | algorithms [COmparin... | numbbo/numbbo · Gi...

corresponds to the master branch as linked above.

3. In a system shell, `cd` **into** the `coco` or `coco-<version>` folder (framework root), where the file `do.py` can be found. Type, i.e. **execute**, one of the following commands once

```
python do.py run-c
python do.py run-java
python do.py run-matlab
python do.py run-octave
python do.py run-python
```

**installation I & test**

depending on which language shall be used to run the experiments. `run-*` will build the respective code and run the example experiment once. The build result and the example experiment code can be found under `code-experiments/build/<language>` ( `<language>=matlab` for Octave). `python do.py` lists all available commands.

4. On the computer where experiment data shall be post-processed, run

```
python do.py install-postprocessing
```

**installation II (post-processing)**

to (user-locally) install the post-processing. From here o... the builds to a new release.

5. **Copy** the folder `code-experiments/build/YOUR-FAVORITE-LANGUAGE` and its content to another location. In Python it is sufficient to copy the file `example_experiment.py`. Run the example experiment (it already is compiled, in case). As the details vary, see the respective read-me's and/or example experiment files:

   - `C` read me and example experiment
   - `Java` read me and example experiment
   - `Matlab/Octave` read me and example experiment

# https://github.com/numbbo/coco

to (user-locally) install the post-processing. From here on, `do.py` has done its job and is only needed again for updating the builds to a new release.

5. **Copy** the folder `code-experiments/build/YOUR-FAVORITE-LANGUAGE` and its content to another location. In Python it is sufficient to copy the file `example_experiment.py`. Run the example experiment (it already is compiled, in case). As the details vary, see the respective read-me's and/or

   **example experiment**

   - `C` read me and example experiment
   - `Java` read me and example experiment
   - `Matlab/Octave` read me and example experiment
   - `Python` read me and example experiment`

   If the example experiment runs, **connect** your favorite algorithm to Coco: replace the call to the random search optimizer in the example experiment file by a call to your algorithm (see above). **Update** the output `result_folder`, the `algorithm_name` and `algorithm_info` of the observer options in the example experiment file.

   Another entry point for your own experiments can be the `code-experiments/examples` folder.

6. Now you can **run** your favorite algorithm on the `bbob-biobj` (for multi-objective algorithms) or on the `bbob` suite (for single-objective algorithms). Output is automatically generated in the specified data `result_folder`.

7. **Postprocess** the data from the results folder by typing

```
python -m bbob_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

   The name `bbob_pproc` will become `cocopp` in future. Any subfolder in the folder arguments will be searched for logged data. That is, experiments from different batches can be in different folders collected under a single "root" `YOURDATAFOLDER` folder. We can also compare more than one algorithm by specifying several data result folders generated by different algorithms.

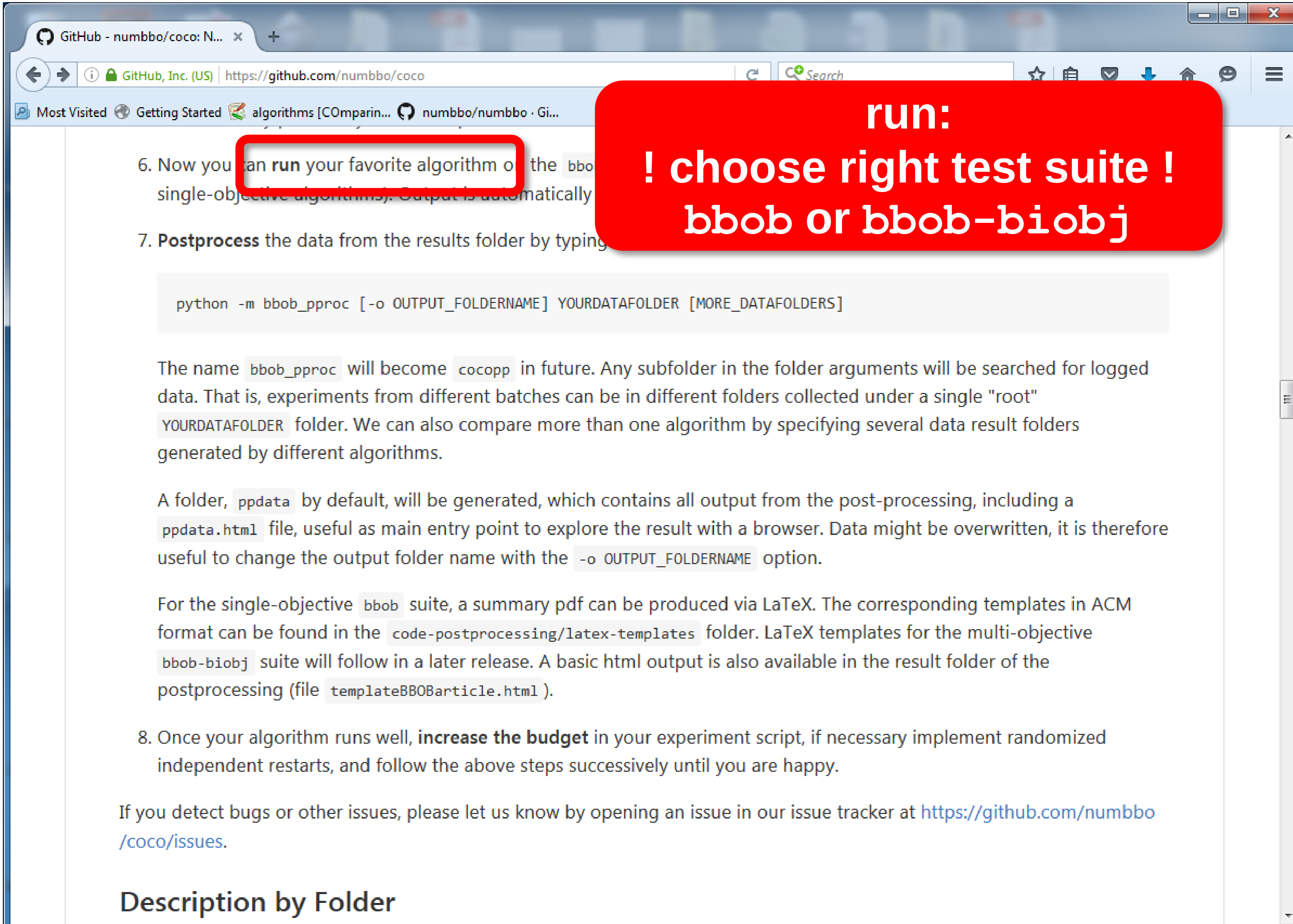
# example_experiment.c

```
/* Iterate over all problems in the suite */
while ((PROBLEM = coco_suite_get_next_problem(suite, observer)) != NULL)
{
    size_t dimension = coco_problem_get_dimension(PROBLEM);

    /* Run the algorithm at least once */
    for (run = 1; run <= 1 + INDEPENDENT_RESTARTS; run++) {

      size_t evaluations_done = coco_problem_get_evaluations(PROBLEM);
      long evaluations_remaining =
        (long)(dimension * BUDGET_MULTIPLIER) - (long)evaluations_done;

      if (... || (evaluations_remaining <= 0))
        break;

      my_random_search(evaluate_function, dimension,
                       coco_problem_get_number_of_objectives(PROBLEM),
                       coco_problem_get_smallest_values_of_interest(PROBLEM),
                       coco_problem_get_largest_values_of_interest(PROBLEM),
                       (size_t) evaluations_remaining,
                       random_generator);
}
```

generated by different algorithms.

# https://github.com/numbbo/coco

run:
! choose right test suite !
bbob or bbob-biobj



GitHub - numbbo/coco: N...

GitHub, Inc. (US) https://github.com/numbbo/coco

Most Visited  Getting Started  algorithms [COmparin...  numbbo/numbbo · Gi...

6. Now you can **run** your favorite algorithm on the bbo... single-objective algorithms). Output is automatically...

7. **Postprocess** the data from the results folder by typing...

```
python -m bbob_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

The name `bbob_pproc` will become `cocopp` in future. Any subfolder in the folder arguments will be searched for logged data. That is, experiments from different batches can be in different folders collected under a single "root" `YOURDATAFOLDER` folder. We can also compare more than one algorithm by specifying several data result folders generated by different algorithms.

A folder, `ppdata` by default, will be generated, which contains all output from the post-processing, including a `ppdata.html` file, useful as main entry point to explore the result with a browser. Data might be overwritten, it is therefore useful to change the output folder name with the `-o OUTPUT_FOLDERNAME` option.

For the single-objective `bbob` suite, a summary pdf can be produced via LaTeX. The corresponding templates in ACM format can be found in the `code-postprocessing/latex-templates` folder. LaTeX templates for the multi-objective `bbob-biobj` suite will follow in a later release. A basic html output is also available in the result folder of the postprocessing (file `templateBBOBarticle.html`).

8. Once your algorithm runs well, **increase the budget** in your experiment script, if necessary implement randomized independent restarts, and follow the above steps successively until you are happy.

If you detect bugs or other issues, please let us know by opening an issue in our issue tracker at https://github.com/numbbo/coco/issues.

## Description by Folder

https://github.com/numbbo/coco

6. Now you can **run** your favorite algorithm on the `bbob-biobj` (for multi-objective algorithms) or on the `bbob` suite (for single-objective algorithms). Output is automatically generated in the specified data `result_folder`.

7. **Postprocess** the data from the results folder by typing

```
python -m bbob_pproc [-o OUTPUT_FOLDERNAME] YOURDATAFOLDER [MORE_DATAFOLDERS]
```

**postprocess**

The name `bbob_pproc` will become `cocopp` in future. Any subfolder in the folder argu… data. That is, experiments from different batches can be in different folders collected u… `YOURDATAFOLDER` folder. We can also compare more than one algorithm by specifying several data result folders generated by different algorithms.

A folder, `ppdata` by default, will be generated, which contains all output from the post-processing, including a `ppdata.html` file, useful as main entry point to explore the result with a browser. Data might be overwritten, it is therefore useful to change the output folder name with the `-o OUTPUT_FOLDERNAME` option.

For the single-objective `bbob` suite, a summary pdf can be produced via LaTeX. The corresponding templates in ACM form… bbo… pos…

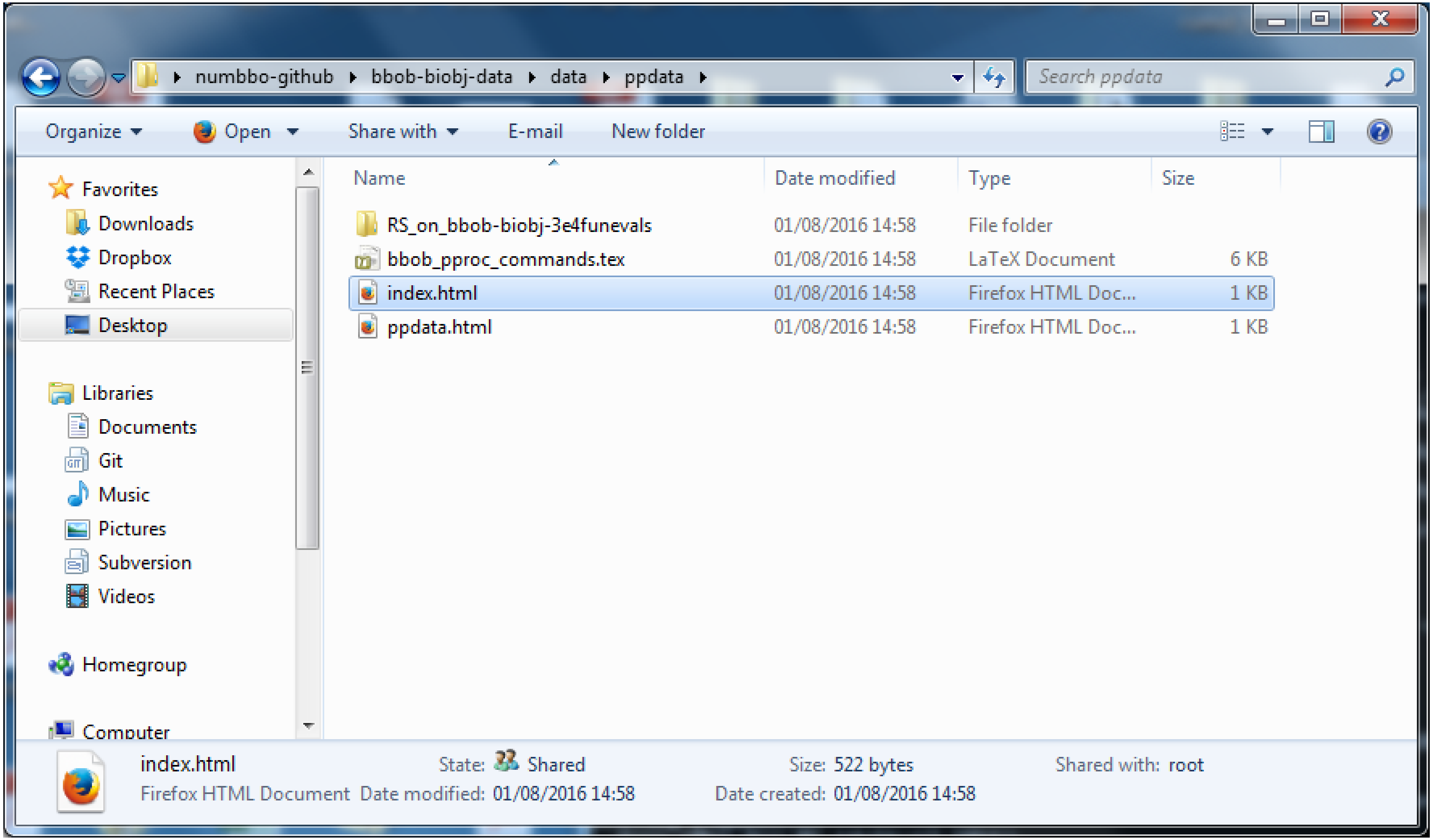
8. On… ind…

If you d… /coco/is…

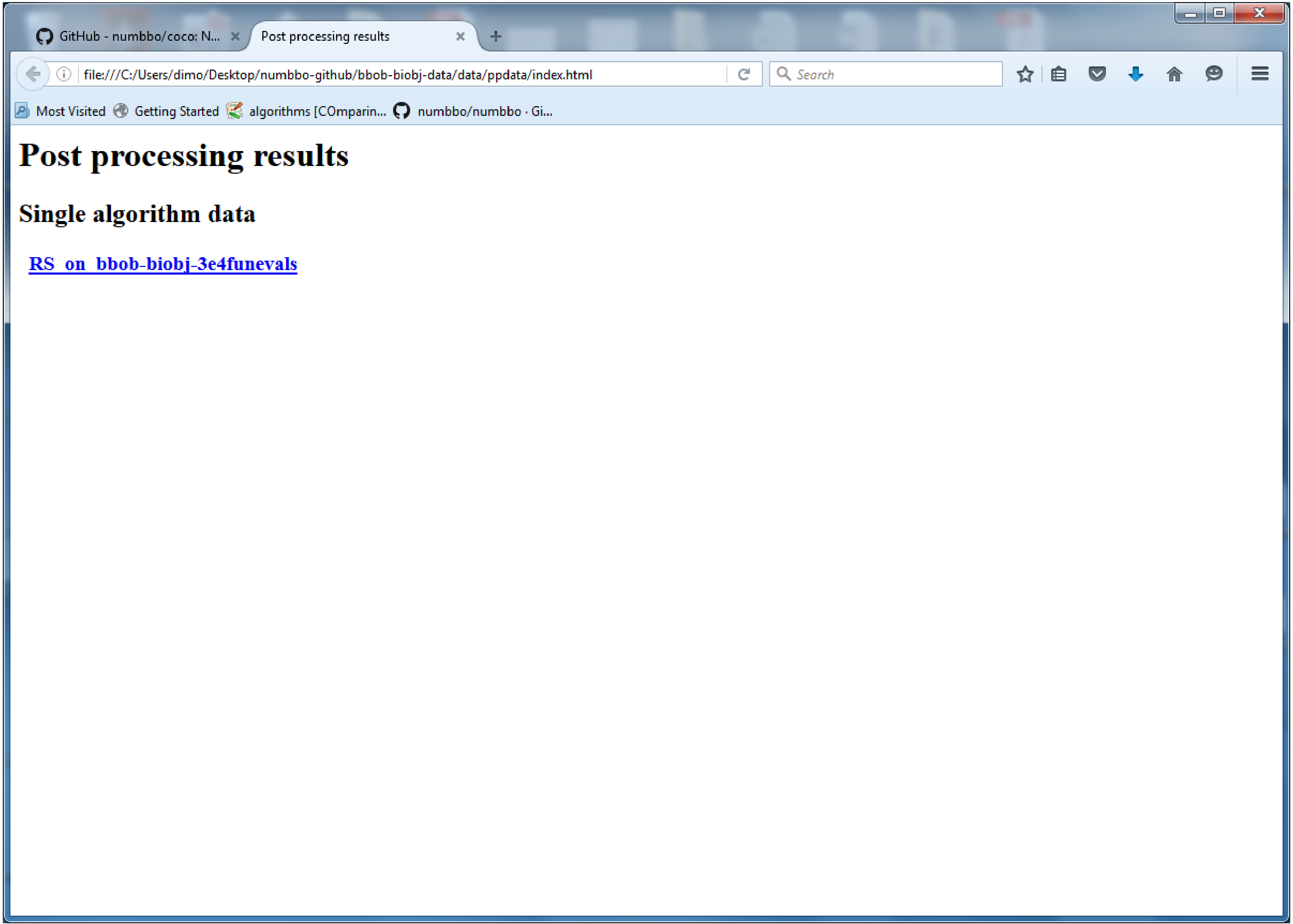
**tip:**
**start with small #funevals (until bugs fixed ☺)**
**then increase budget to get a feeling**
**how long a "long run" will take**

## Description by Folder

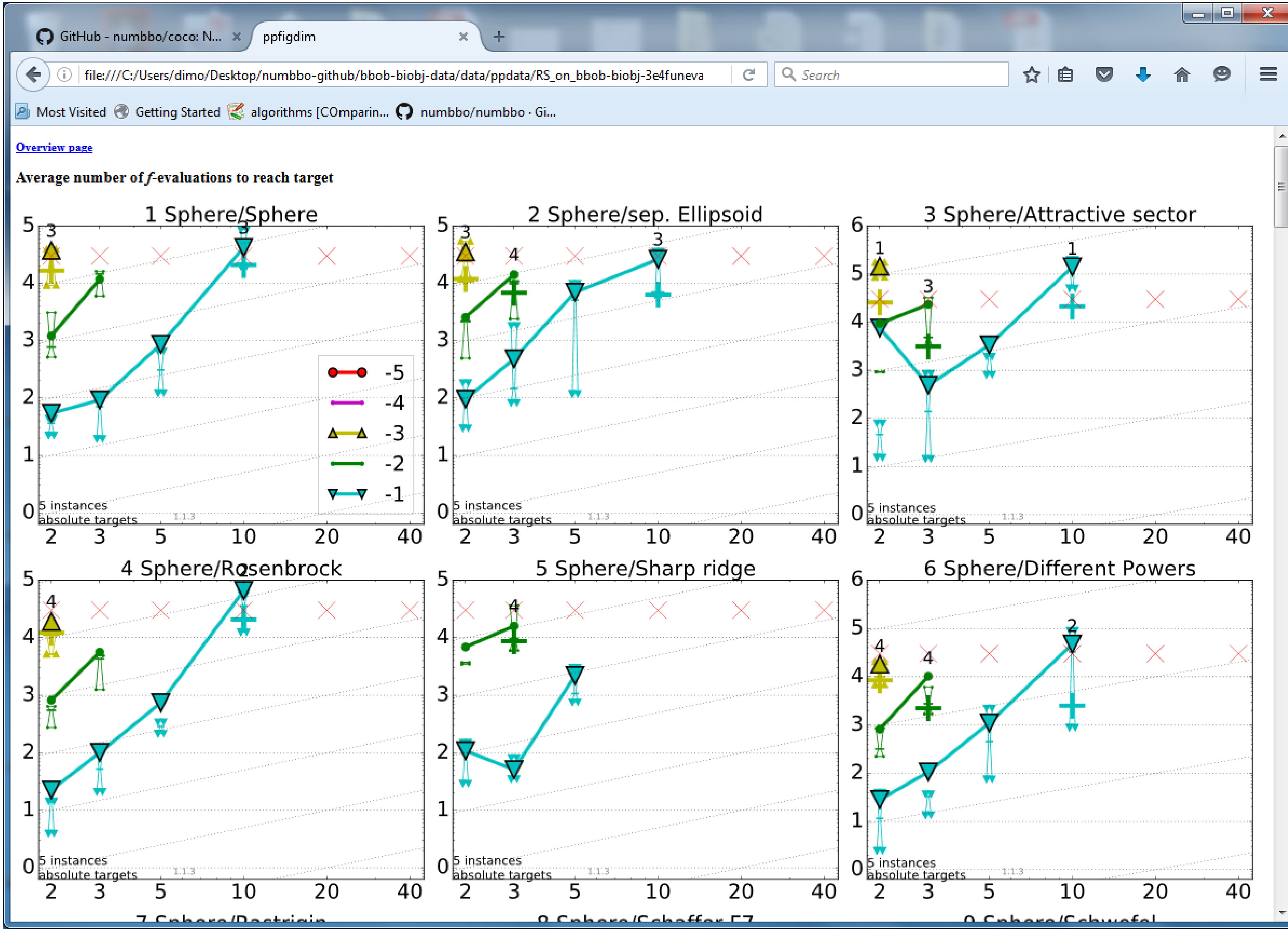
# result folder



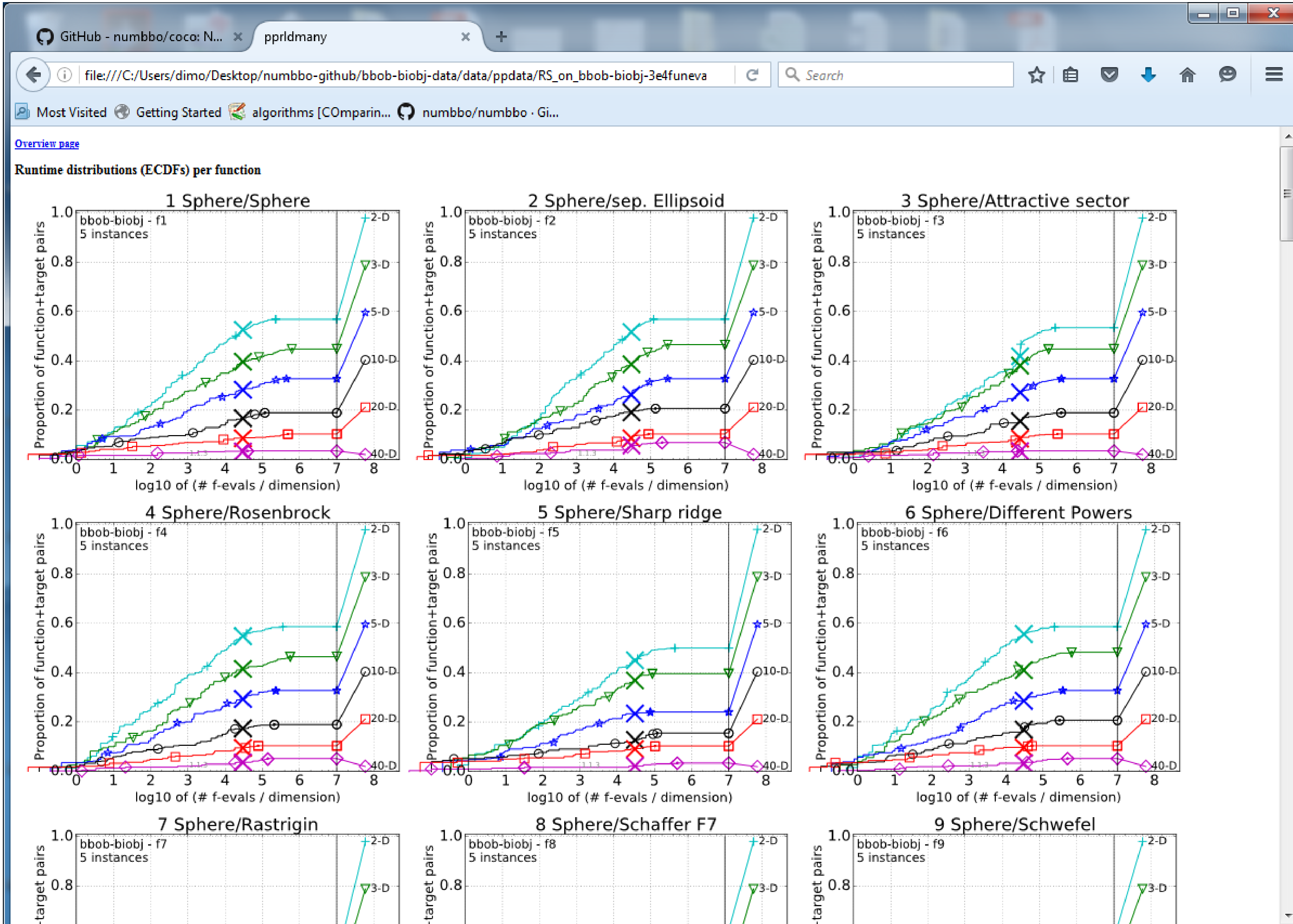
# automatically generated results



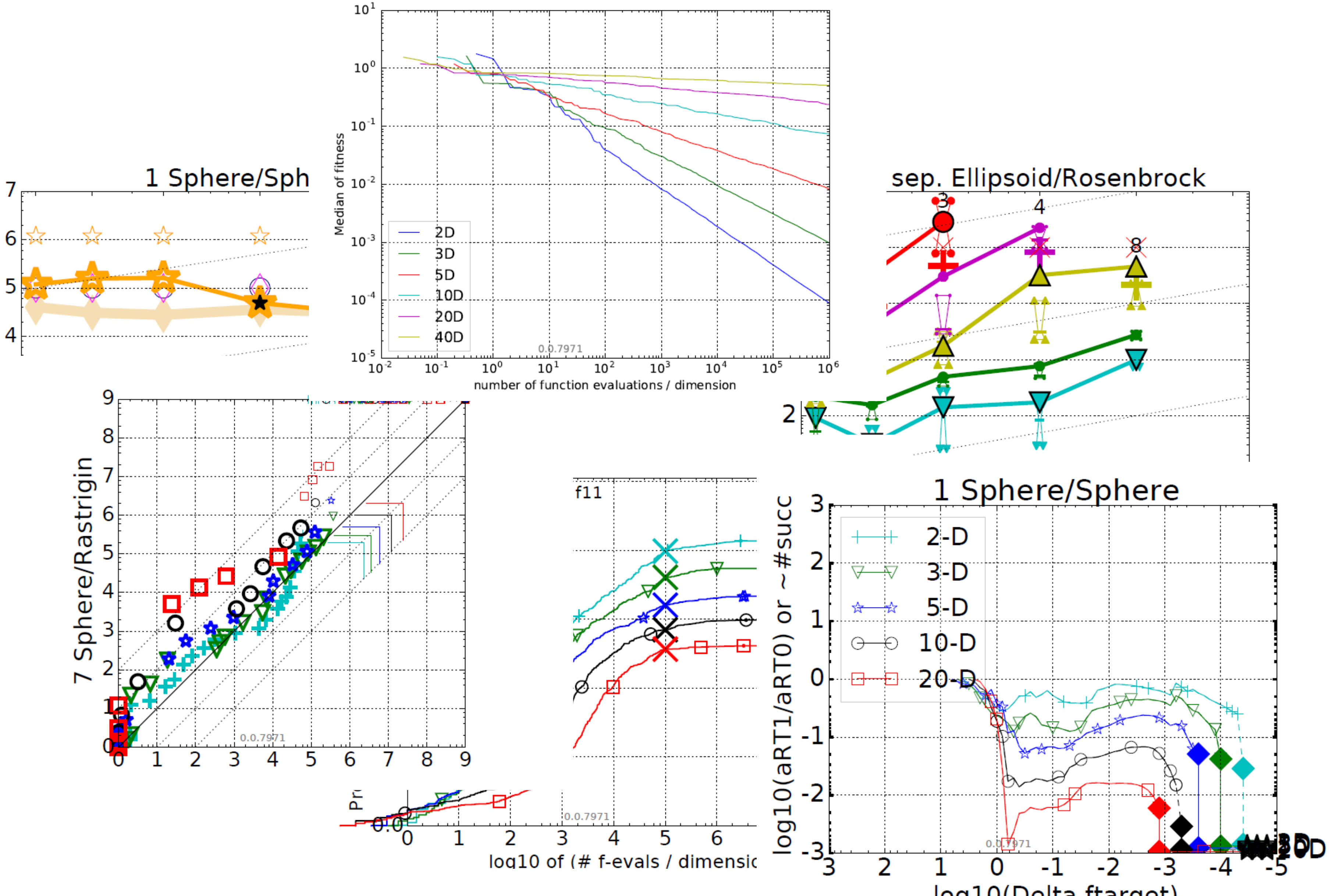
# automatically generated results



# automatically generated results



# More Automated Plots...



**doesn't look too complicated, does it?**

[the devil is in the details ☺]

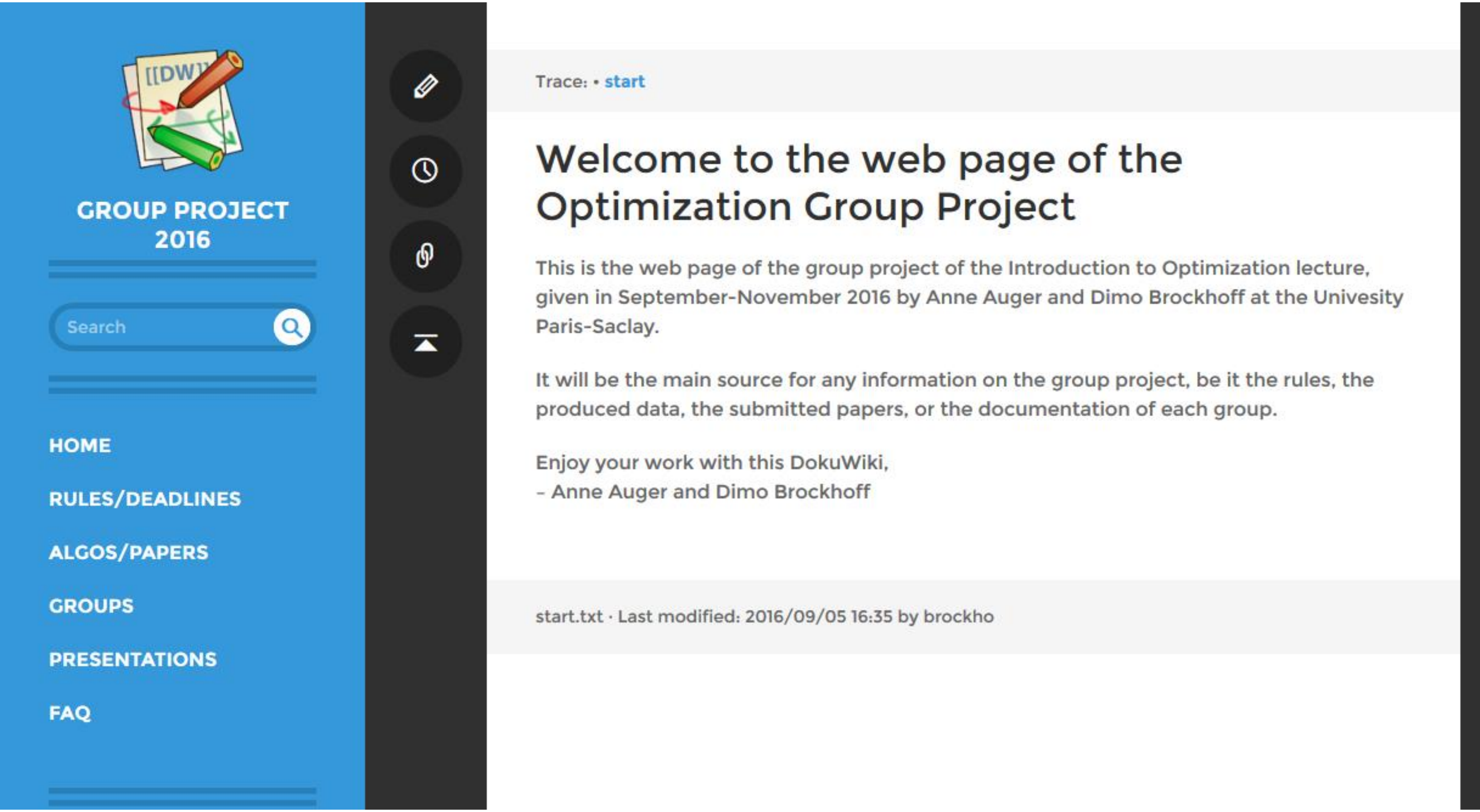
# Course Overview

| | | |
|---|---|---|
| 1 | Fri, 16.9.2016 | Today's lecture: more infos in the end |
| | Wed, 21.9.2016 | **groups defined via wiki** |
| | Thu, 22.9.2016 | everybody went (actively!) through the Getting Started part of github.com/numbbo/coco |
| 2 | Fri, 23.9.2016 | Lecture, final adjustments of groups<br>everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture) |
| 3 | Fri, 30.9.2016 | Lecture |
| 4 | Fri, 7.10.2016 | Lecture |
| | Mon, 10.10.2016 | deadline for intermediate wiki report:<br>what has been done and what remains to be done? |
| 5 | Fri, 14.10.2016 | Lecture |
| 6 | Tue, 18.10.2016 | Lecture |
| | Tue, 18.10.2016 | deadline for submitting data sets |
| | Fri, 21.10.2016 | deadline for paper submission |
| | | vacation |
| 7 | Fri, 4.11.2016 | Final lecture |
| | 7.-11.11.2016 | oral presentations (individual time slots) |
| | 14 - 18.11.2016 | Exam (exact date to be confirmed) |

All deadlines:
23:59pm Paris time

# Group Project Wiki

http://randopt.gforge.inria.fr/teaching/optimization-Saclay/groupproject2016/



# Group Project Wiki

- to be found at
  - http://randopt.gforge.inria.fr/teaching/optimization-Saclay/groupproject2016/
  - also via a link on the home page
- please use this to interact within the groups
  - document what you do
  - document who is doing what
  - document what still needs to be done (intermediate report due on Monday October 10, 2016)
- and coordinate the assignments of all of you to groups with paper/algorithm and programming language (by next Wednesday!)
  - 6 algorithms available
  - 0, 1, or 2 groups per algorithm
  - if 2 groups: choose different programming language!

    easiest: choose among python, C/C++, Java, Matlab/Octave

# Course Overview

| | | |
|---|---|---|
| 1 | Fri, 16.9.2016 | Today's lecture: more infos in the end |
| | Wed, 21.9.2016 | groups defined via wiki |
| | Thu, 22.9.2016 | **everybody went (actively!) through the Getting Started part of github.com/numbbo/coco** |
| 2 | Fri, 23.9.2016 | Lecture, final adjustments of groups<br>everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture) |
| 3 | Fri, 30.9.2016 | Lecture |
| 4 | Fri, 7.10.2016 | Lecture |
| | Mon, 10.10.2016 | deadline for intermediate wiki report:<br>what has been done and what remains to be done? |
| 5 | Fri, 14.10.2016 | Lecture |
| 6 | 17./18.10.2016 | Lecture |
| | Tue, 18.10.2016 | deadline for submitting data sets |
| | Fri, 21.10.2016 | deadline for paper submission |
| | | vacation |
| 7 | Fri, 4.11.2016 | Final lecture |
| | 7.-11.11.2016 | oral presentations (individual time slots) |
| | 14 - 18.11.2016 | Exam (exact date to be confirmed) |

All deadlines:
23:59pm Paris time

# Presentation
# Blackbox Optimization
# Lecture

# Presentation Black Box Optimization Lecture

- Optional class “Black Box Optimization”
- Taught also by Anne Auger and me
- Advanced class, (even) closer to our actual research topic

**Goals:**

➊ present the latest knowledge on blackbox optimization algorithms and their foundations

➋ offer hands-on exercises on difficult common optimization problems

➌ give insights into what are current challenging research questions in the field of blackbox optimization (as preparation for a potential Master’s or PhD thesis in the field)

- ☺ relatively young research field with many interesting research questions (in both theory and algorithm design)
- ☺ related to real-world problems: also good for a job outside academia

# Black Box Scenario

$$x \in \Omega \quad \longrightarrow \quad \boxed{\textbf{black box}} \quad \longrightarrow \quad \mathcal{F}(x) \in \mathbb{R}$$

**Why are we interested in a black box scenario?**

- objective function $\mathcal{F}$ often noisy, non-differentiable, or sometimes not even understood or available
- objective function $\mathcal{F}$ contains legacy or binary code, is based on numerical simulations or real-life experiments
- most likely, you will see such problems in practice...

**Objective:** find $x$ with small $\mathcal{F}(x)$ with as few function evaluations as possible

*assumption: internal calculations of algo irrelevant*

# What Makes an Optimization Problem Difficult?

- Search space too large

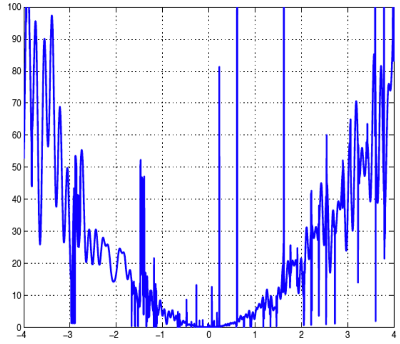
  *exhaustive search impossible*

- Non conventional objective function or search space

  *mixed space, function that cannot be computed*

- Complex objective function

  *non-smooth, non differentiable, noisy, ...*




**stochastic search algorithms**
well suited because they:

- don't make many assumptions on $\mathcal{F}$
- are invariant wrt. translation/rotation of the search space, scaling of $\mathcal{F}$, ...
- are robust to noise

# Planned Topics / Keywords

- Introduction to stochastic search algorithms, in particular
  - Evolutionary algorithms
  - Evolution Strategies and the CMA-ES algorithm in depth
  - Algorithms for large-scale problems (“big data”)
- Multiobjective optimization
- In more detail: Benchmarking black box algorithms

- Combination of lectures & exercises, theory & practice
- Connections with machine learning class of M. Sebag

# Conclusions Greedy Algorithms II

I hope it became clear...

...what kind of optimization problems we are interested in

...what are the requirements for the group project and the exam

...and what are the next important steps to do:

by Wednesday: build the groups and decide on an algorithm

by Thursday: go through the "Getting Started" of COCO