Introduction to Optimization

September 18, 2017

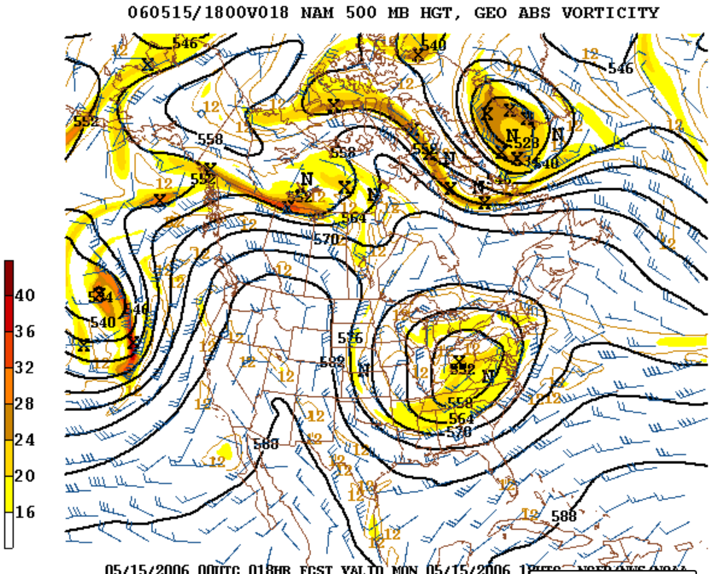
TC2 - Optimisation

Université Paris-Saclay, Orsay, France

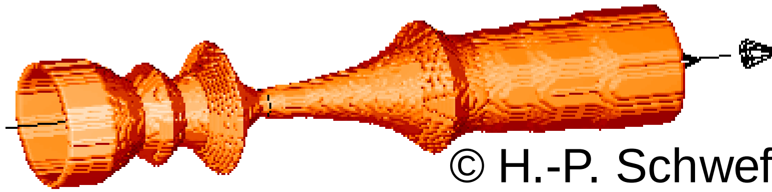
Dimo Brockhoff

Inria Saclay – Ile-de-France

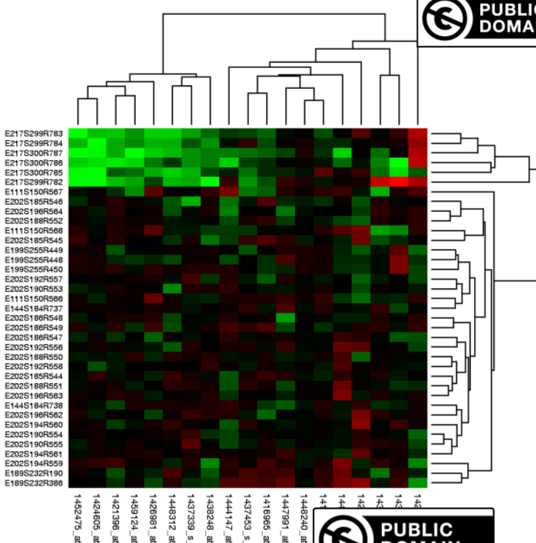
What is Optimization?



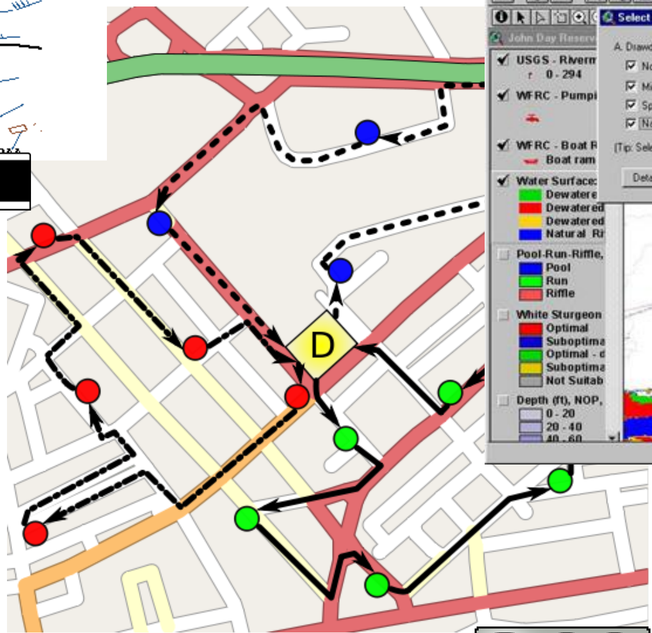
PUBLIC DOMAIN



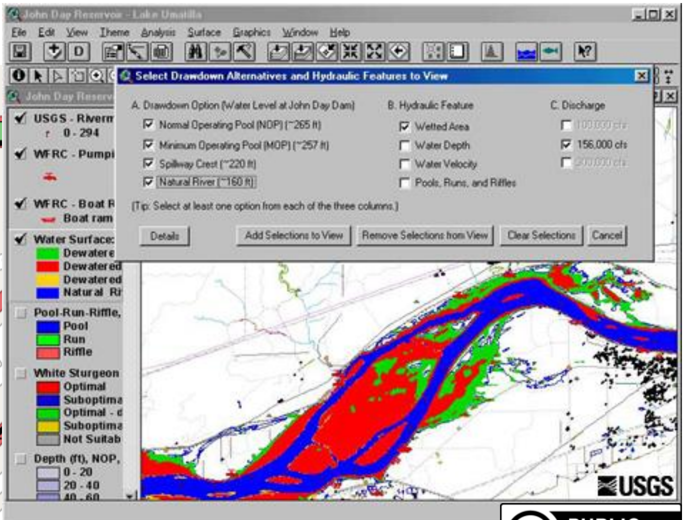
© H.-P. Schwefel



PUBLIC DOMAIN



Maly LOLek



PUBLIC DOMAIN

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
Mon, 18.9.2017	Introduction and Group Project
Wed, 20.9.2017	Benchmarking with the COCO Platform (Group Project)
Fri, 22.9.2017	Introduction to Continuous Optimization
Fri, 29.9.2017	Gradient-Based Algorithms
Fri, 6.10.2017	Stochastic Algorithms and Derivative-free Optimization
Fri, 13.10.2017	Graph Theory, Greedy Algorithms and Dynamic programming
Fri, 20.10.2017	Dynamic Programming, Branch and Bound and Heuristics
vacation	
Fri, 10.11.2017	Exam

all classes + exam are from 14h till 17h15 (incl. a 15min break)
here in D101 (except for E210 this Wednesday)

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:

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

➋ understand the fundamental concepts of optimization algorithms

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

The Exam

- open book: take as much material as you want
- (most likely) multiple-choice
- Friday, 10th of November 2017

- 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

looks a lot ;-)

but: important to go out of your comfort zone to learn!

Course Overview

1	Mon, 18.9.2017	today's lecture: more infos in the end
	Tue, 19.9.2017	groups defined via wiki
		everybody went (actively!) through the Getting Started part of github.com/numbbo/coco
2	Wed, 20.9.2017	lecture "Benchmarking", final adjustments of groups everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture)
3	Fri, 22.9.2017	lecture "Introduction to Continuous Optimization"
4	Fri, 29.9.2017	lecture "Gradient-Based Algorithms"
5	Fri, 6.10.2017	lecture "Stochastic Algorithms and DFO"
6	Fri, 13.10.2017	lecture "Discrete Optimization I: graphs, greedy algos, dyn. progr." deadline for submitting data sets
	Wed, 18.10.2017	deadline for paper submission
7	Fri, 20.10.2017	final lecture "Discrete Optimization II: dyn. progr., B&B, heuristics"
	Thu, 26.10.2017 / Fri, 27.10.2017	oral presentations (individual time slots)
	after 30.10.2017	vacation aka learning for the exams
	Fri, 10.11.2017	written exam

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://www.cmap.polytechnique.fr/`
`~dimo.brockhoff/optimizationSaclay/2017/`

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

Advertisement: Master's Thesis Topics

RandOpt team
Inria and Ecole Polytechnique

Permanent members:
Anne Auger, Dimo Brockhoff, Nikolaus Hansen
https://team.inria.fr/randopt/team-members/

Master's theses available (PhD theses possible) :

- start anytime
- 6 months
- paid via Inria
- many topics around blackbox optimization
- theory ↔ algorithm design

constrained
large-scale
multiobjective
CMA-ES
blackbox optimization
expensive
theory
applications
algorithm design
benchmarking

http://randopt.gforge.inria.fr/thesisprojects/

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 on Wednesday)

Given:

set of possible solutions — *Search space*

quality criterion — *Objective function*

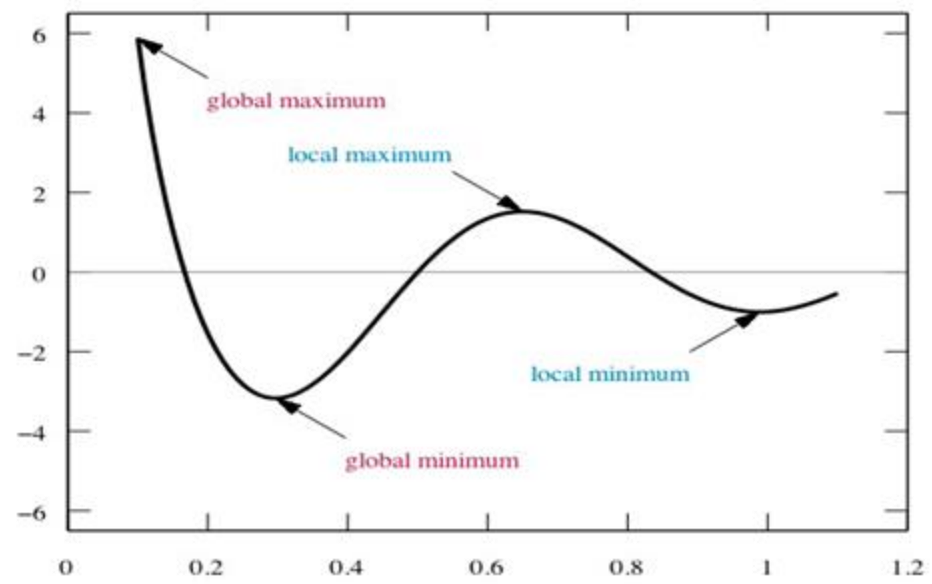
Objective:

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 Ω

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 Ω

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

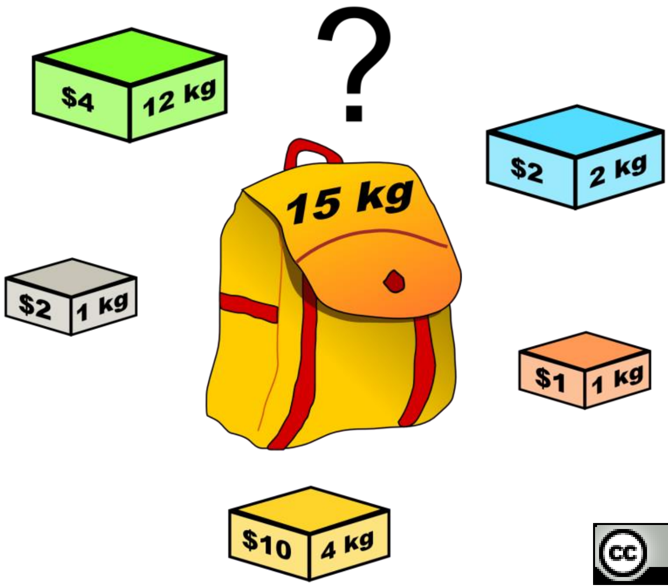
[e.g. constraints related to manufactoring precisions vs. cost constraints]

Example 1: Combinatorial Optimization

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]



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

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

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

Example 2: Combinatorial Optimization

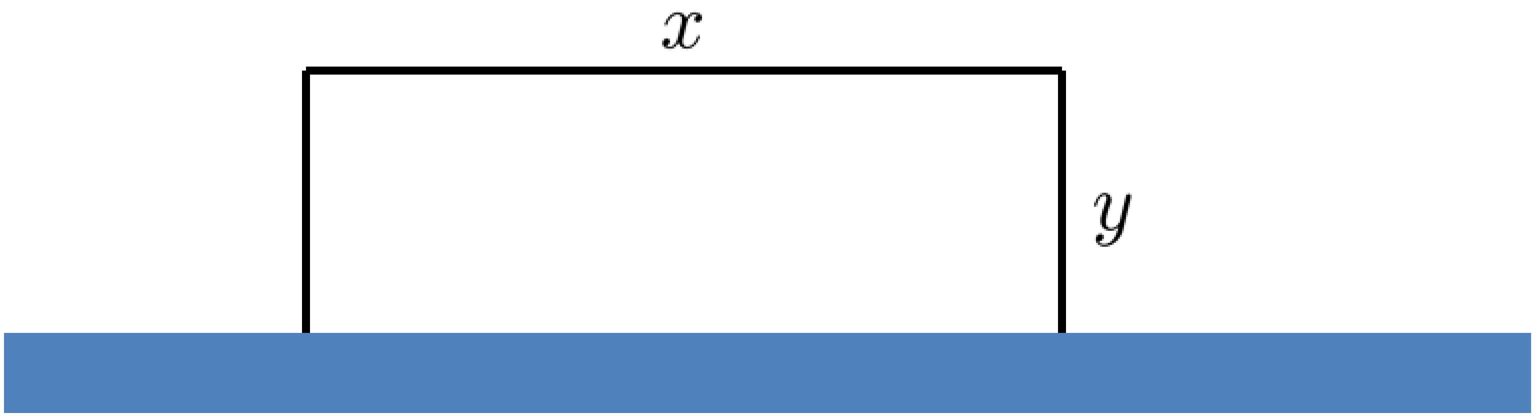
Traveling Salesperson Problem (TSP)

- 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?

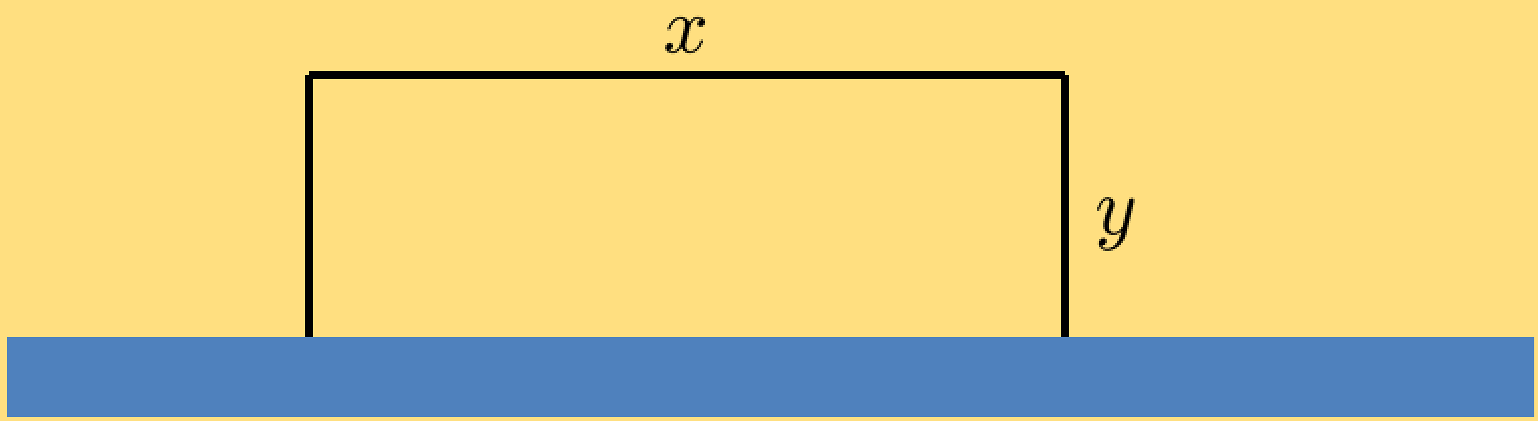


Exercise:

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

Example 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?



solution can be found analytically:

exercise for the weekend ;-)

$\Omega = \mathbb{R}_+^2$:

$\max xy$

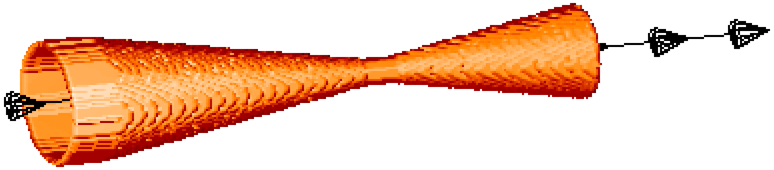
where $x + 2y \leq 500$

Example 4: A "Manual" Engineering Problem

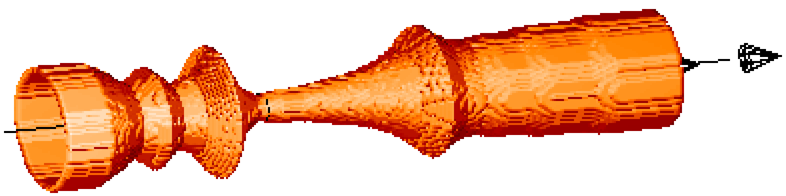
Optimizing a Two-Phase Nozzle [Schwefel 1968+]

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

initial design:



final design:



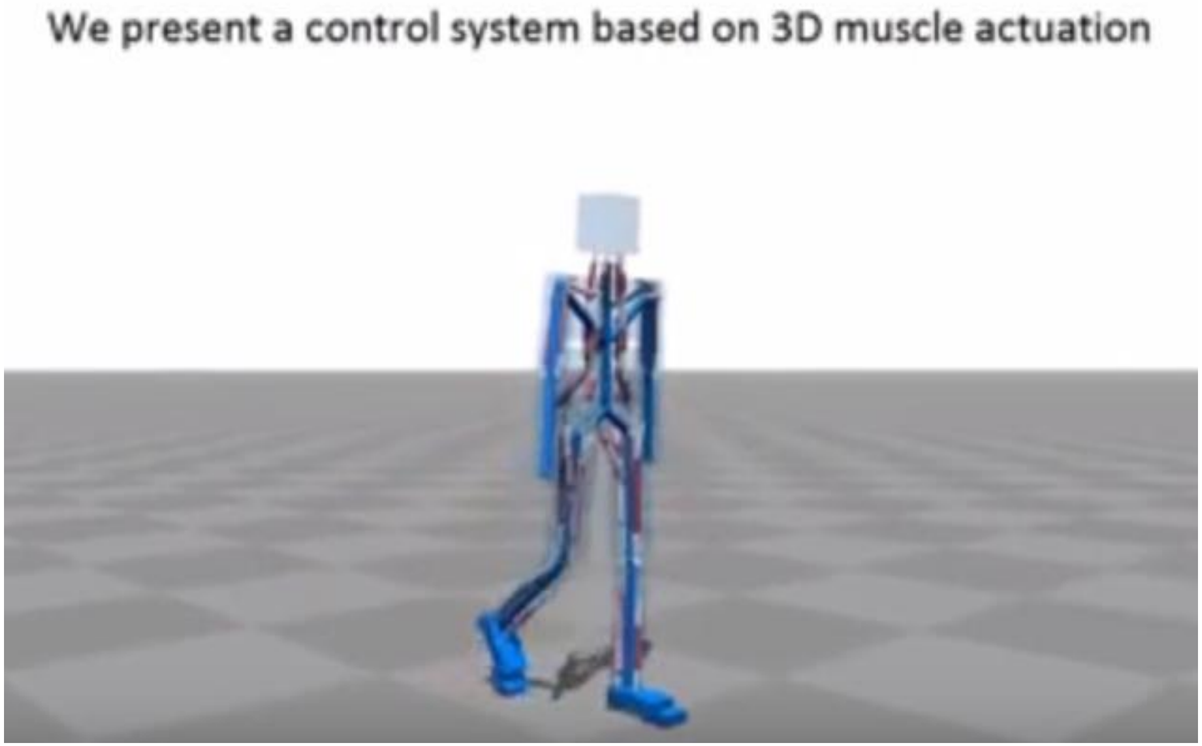
$\Omega =$ all possible nozzles of given number of slices

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:

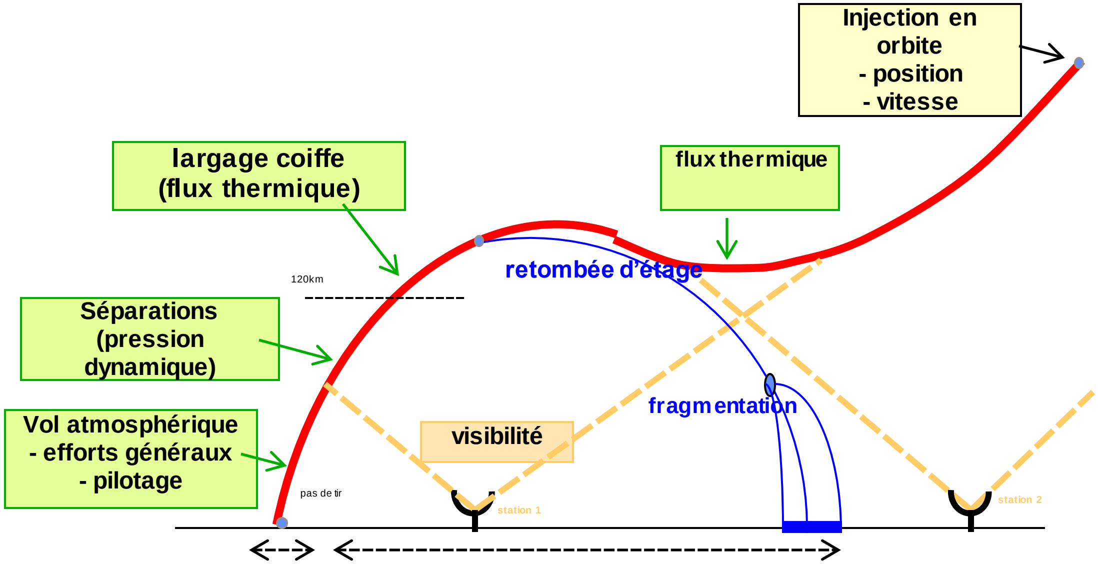


https://www.youtube.com/watch?v=yci5FuI1ovk

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

Design of a Launcher

Poppy



- 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

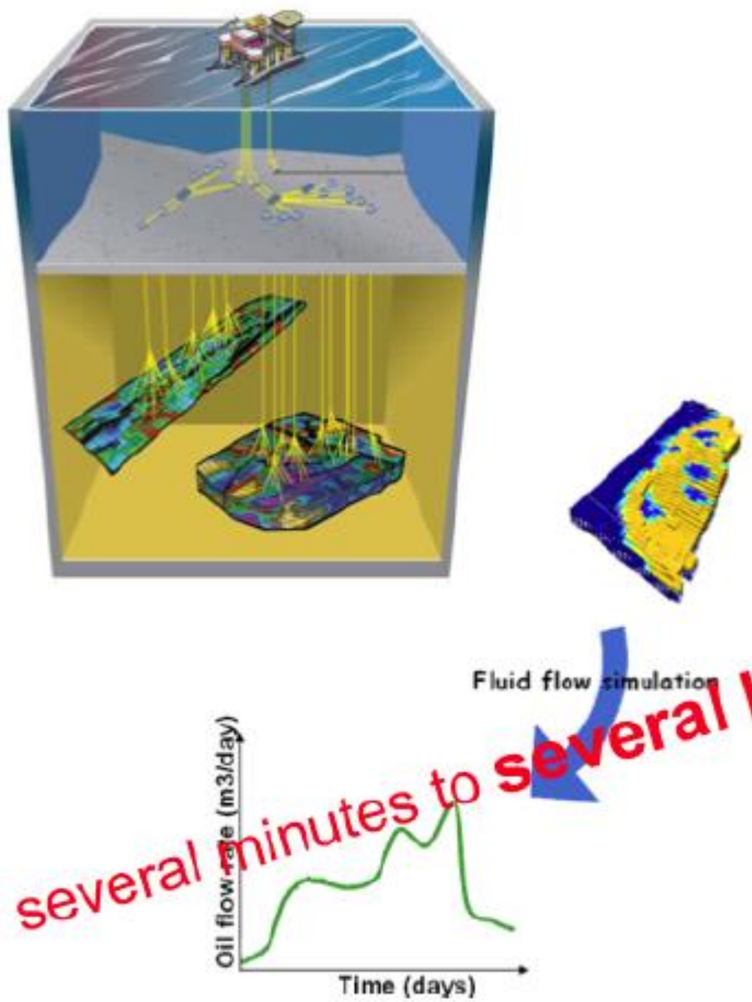
23 continuous parameters to optimize

+ constraints

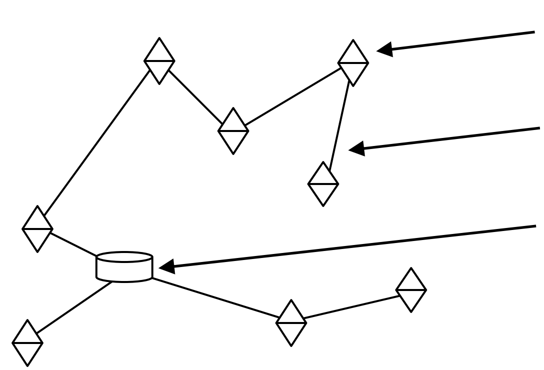
$$\Omega = \mathbb{R}^{23}$$

Example 7: An Expensive Real-World Problem

Well Placement Problem



several minutes to several hours !!



for a given structure,
per well:

- angle & distance to previous well
- well depth

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

$\sigma \in \Omega$: variable length!

Example 8: Data Fitting – Data Calibration

Objective

- Given a sequence of data points $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}, i = 1, \ldots, N$, 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) $\{y_i, \boldsymbol{x}_i\}_{i=1,\ldots,N}$

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

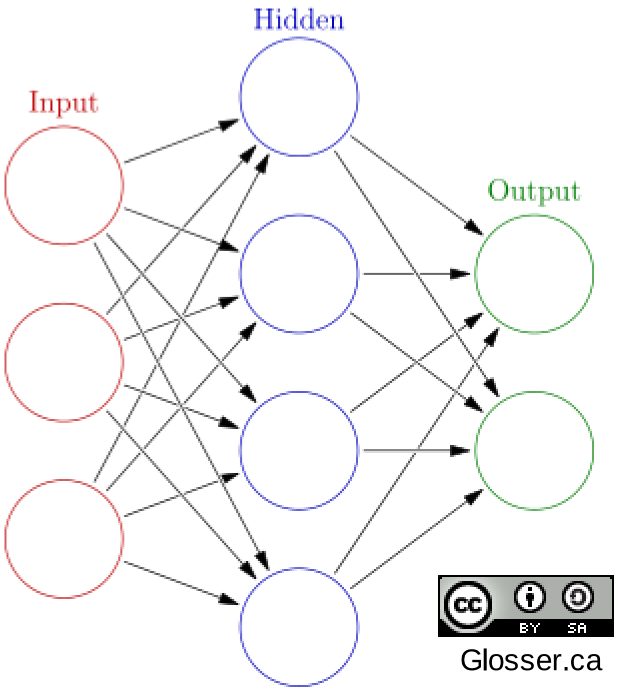
Given a set of data: $\{y_i, \underbrace{x_i^1, \ldots, x_i^p}_{\boldsymbol{x}_i^T}\}_{i=1\ldots N}$

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

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

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 10: Deep Learning



Actually the same idea:

match model best to given data

Model here:

artificial neural nets
with many hidden layers
(aka deep neural networks)

Parameters to tune:

- weights of the connections (continuous parameter)
- topology of the network (discrete)
- firing function (less common)

Example 11: 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
- unconstrained vs. constrained (and then which type of constraint)
- one or multiple objective functions

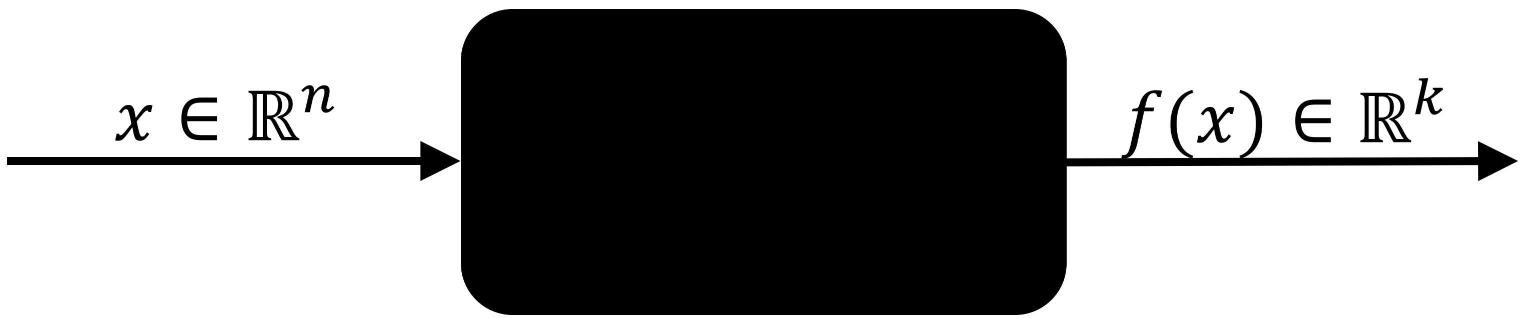
Not covered in this introductory lecture:

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

Example: Numerical Blackbox Optimization

Typical scenario in the continuous, unconstrained 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
  - bound constraints (on the variables only)
  - linear/quadratic/non-linear constraints
  - blackbox constraints
  - many more

(see e.g. Le Digabel and Wild (2015), https://arxiv.org/abs/1505.07881)

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

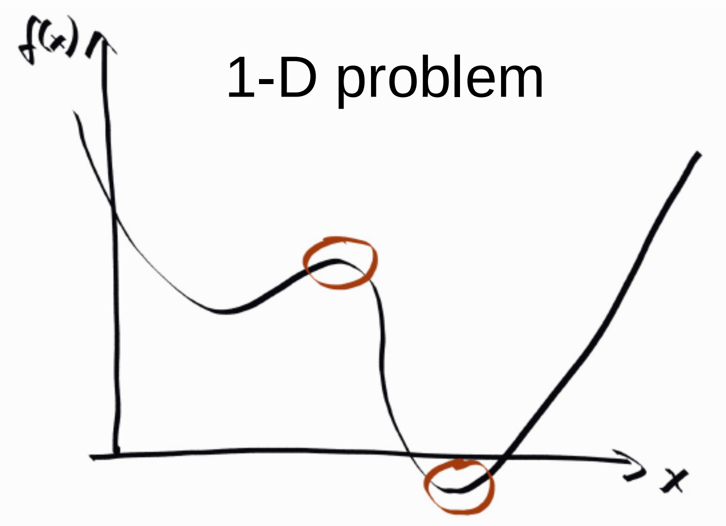
- Optimize f: $\begin{cases} \Omega \subset \mathbb{R}^n \to \mathbb{R} \\ x = (x_1, \dots, x_n) \to f(x_1, \dots, x_n) \end{cases}$

  $\in \mathbb{R}$

  unconstrained optimization

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

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



Unconstrained vs. Constrained Optimization

Unconstrained optimization

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

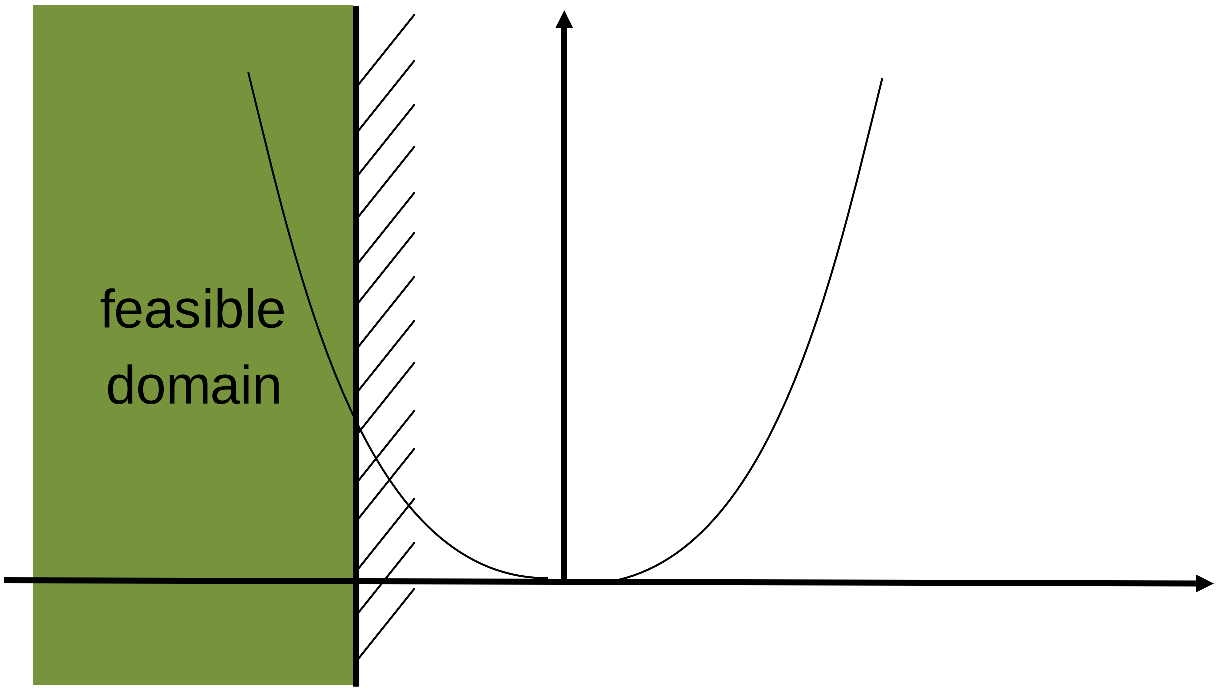
Constrained optimization

- 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)

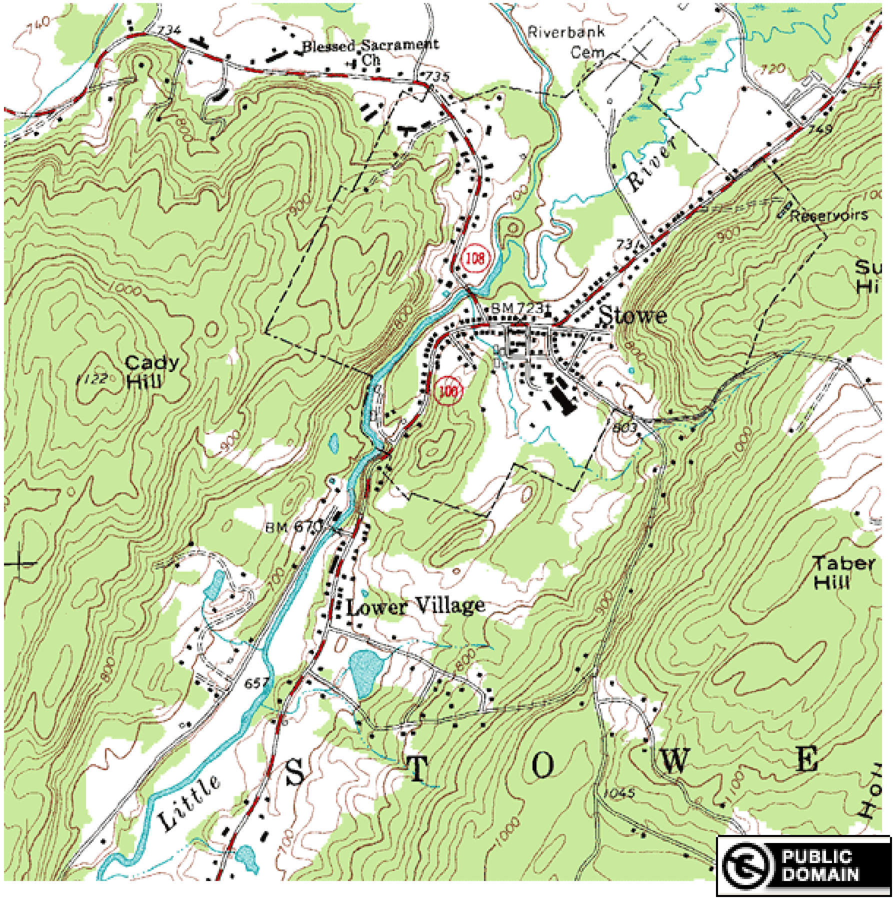
Exercise:
What is the minimum of $f_2(x)$?

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

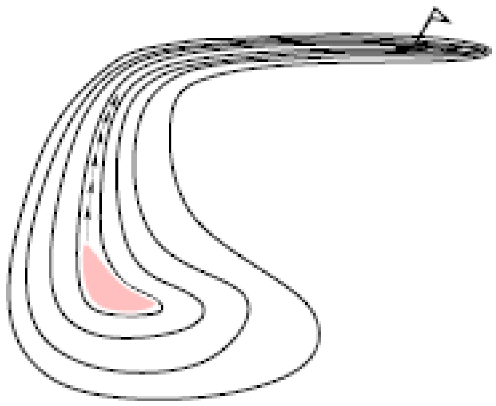
  (considerably) larger than three

- non-separability

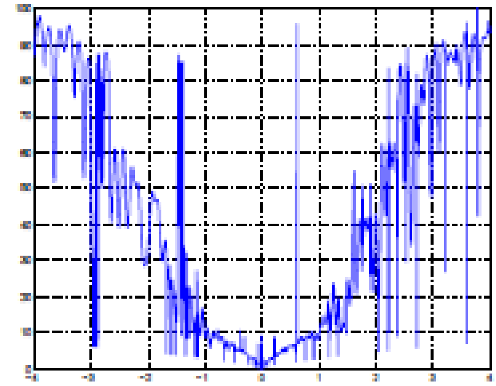
  dependencies between the objective variables

- 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

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

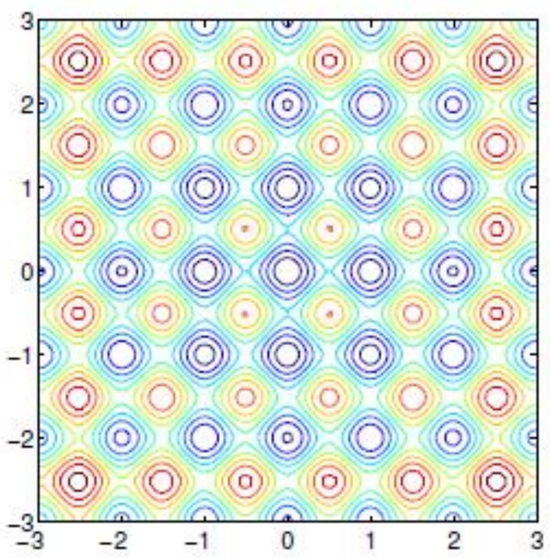
⟹ it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example:

Additively decomposable functions

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

Rastrigin function



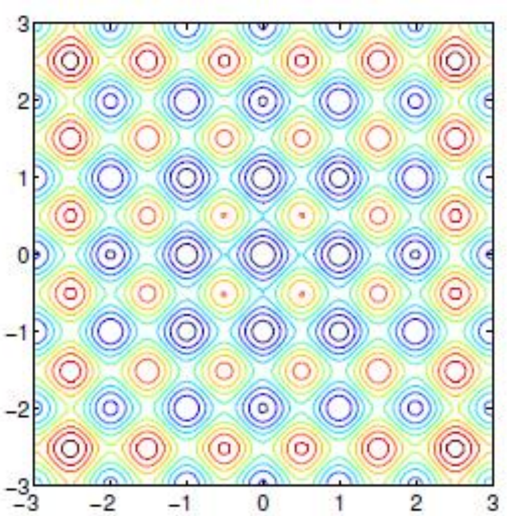
Non-Separable Problems

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

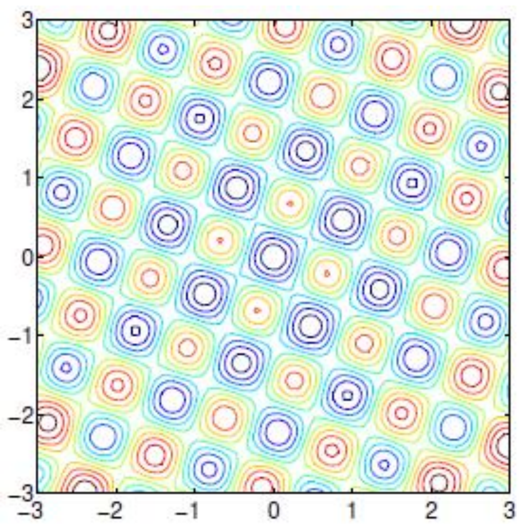
Rotating the coordinate system

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

R rotation matrix



$$\xrightarrow{R}$$



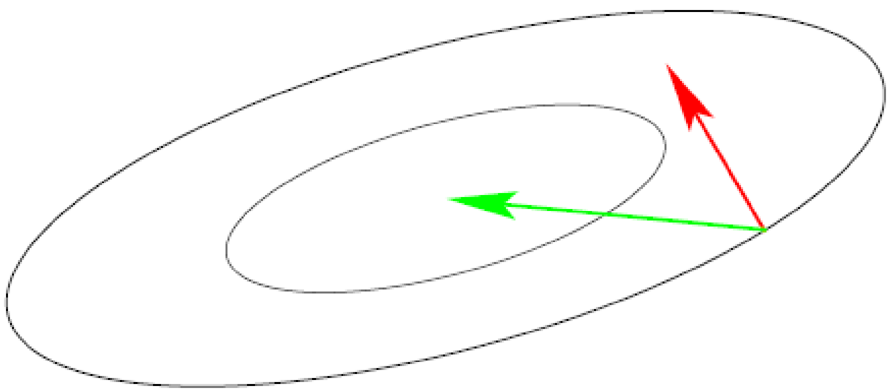
[1] N. Hansen, A. Ostermeier, A. Gawelczyk (1995). "On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation". Sixth ICGA, pp. 57-64, Morgan Kaufmann

[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

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 $\boldsymbol{x}^*$: ∃ 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

Constrained case

- a bit more involved
- hence, later in the lecture ☺

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

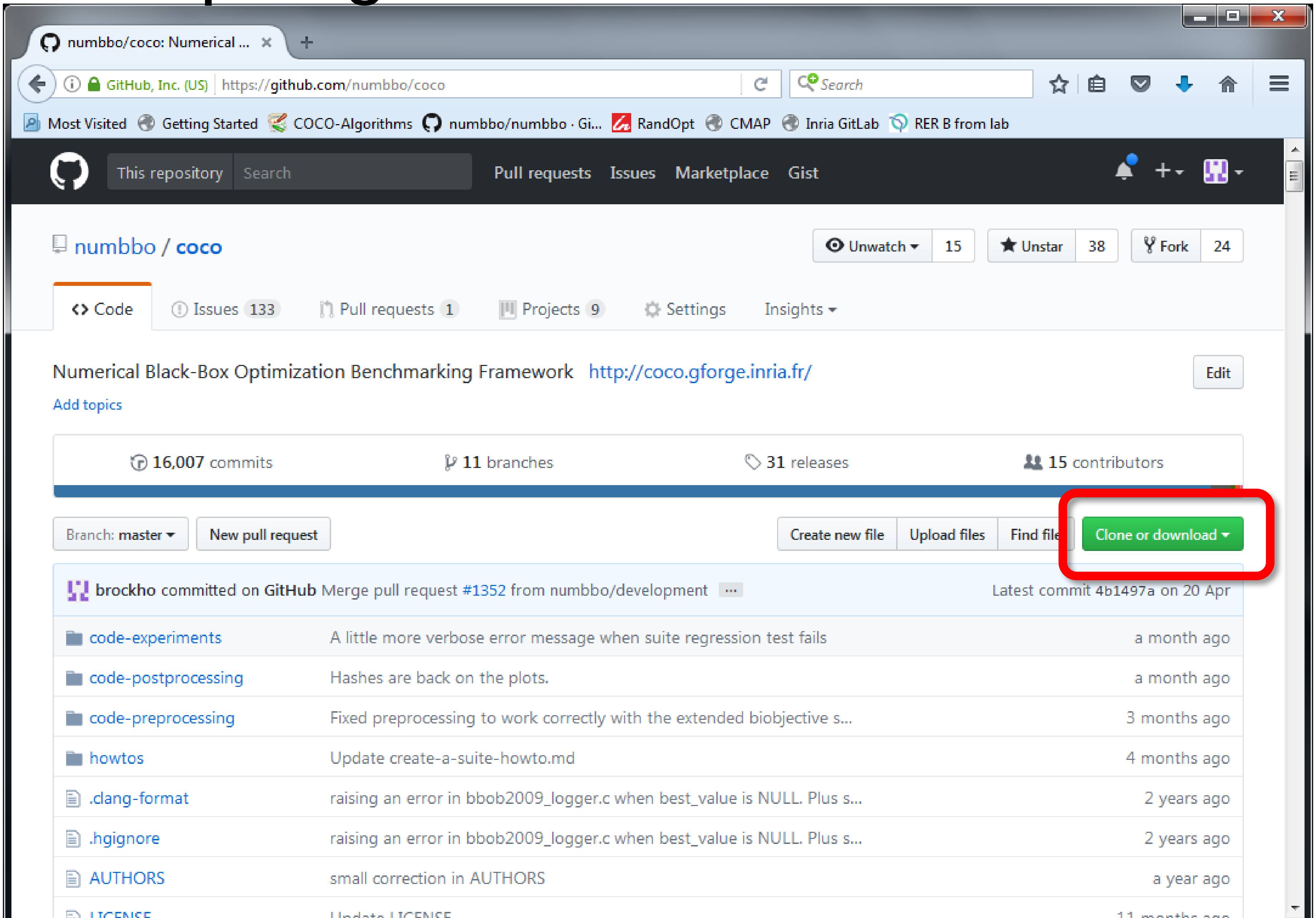
Comparing Continuous Optimizers Platform

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

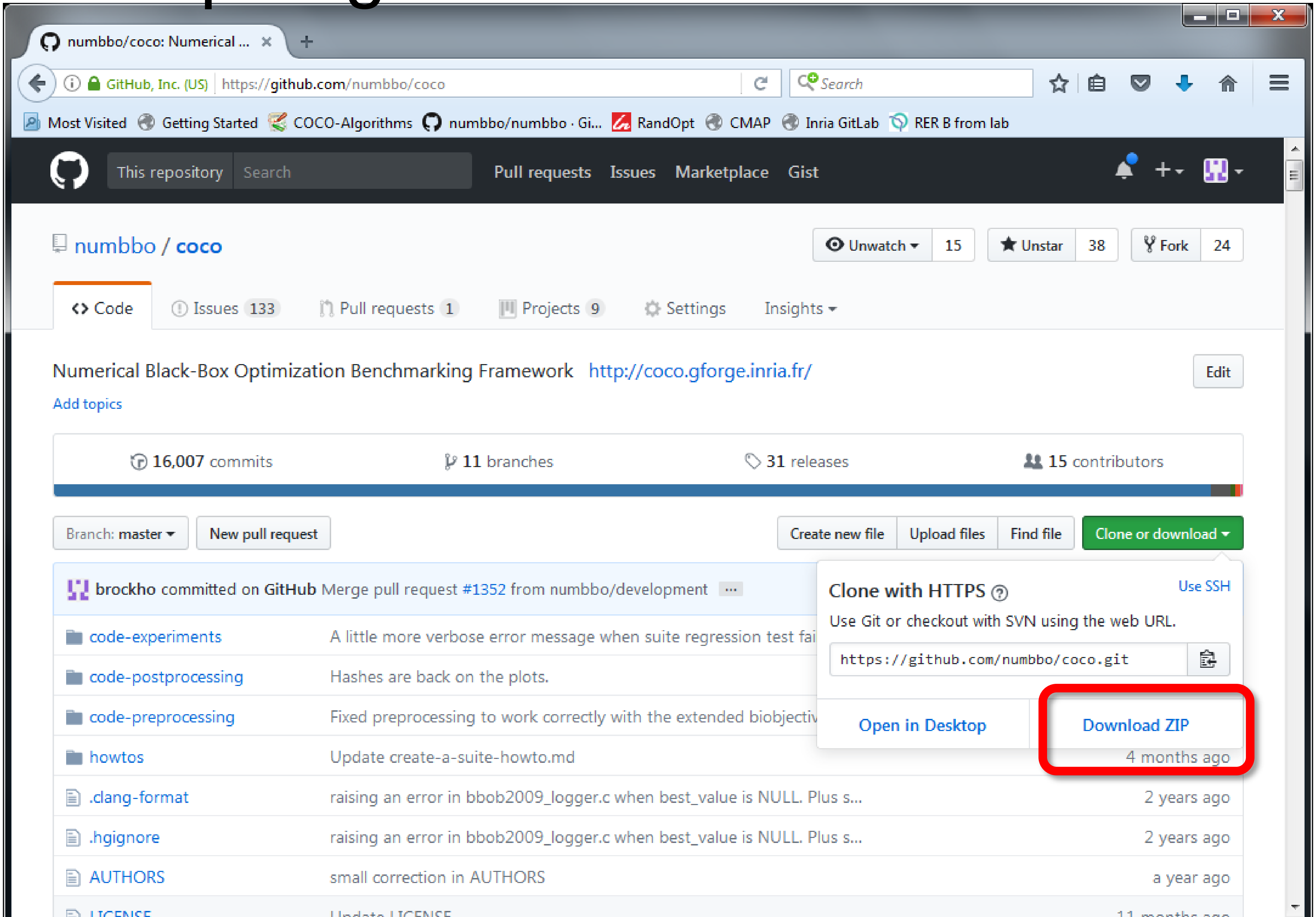
automatized benchmarking

How to benchmark algorithms with COCO?

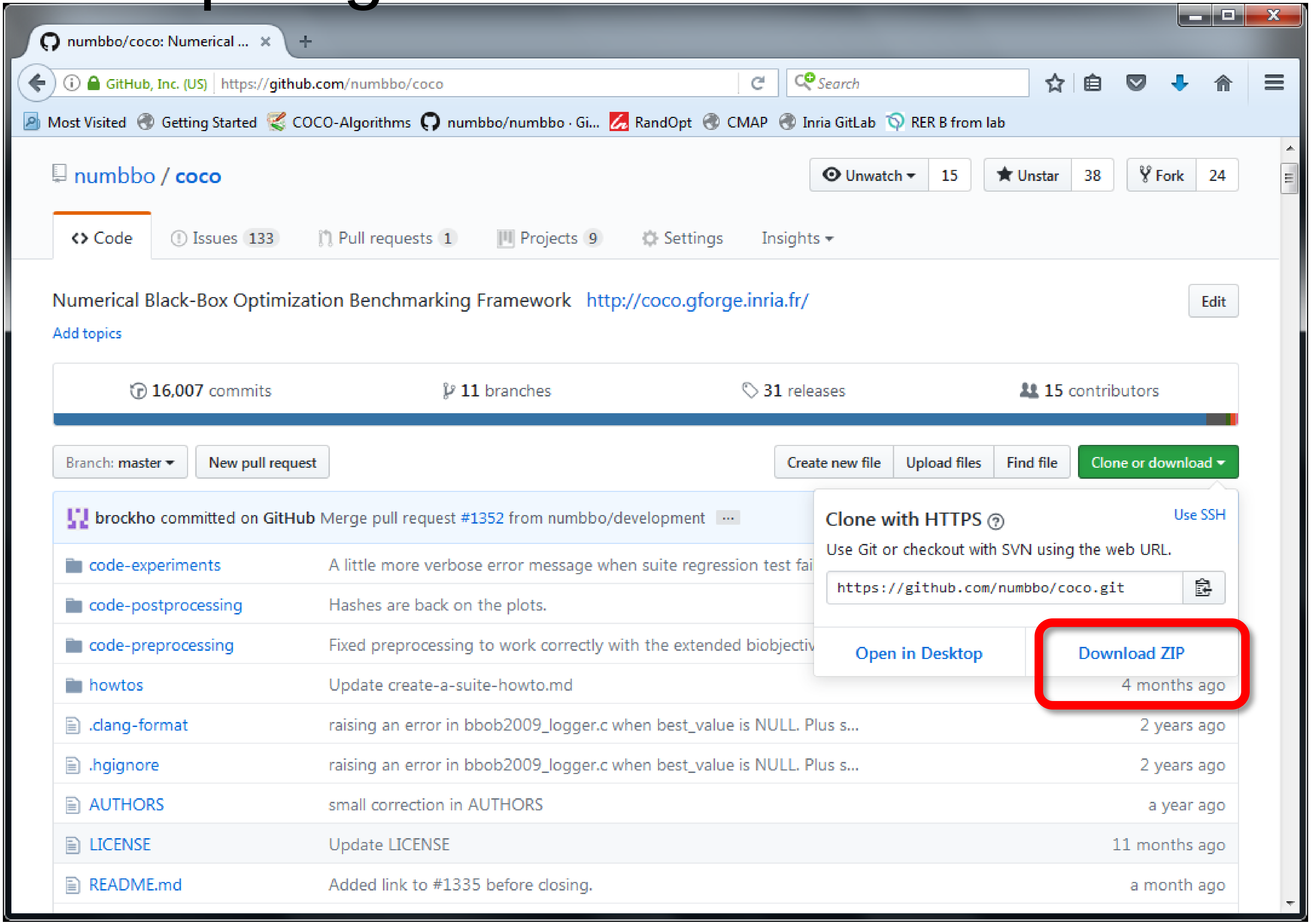
https://github.com/numbbo/coco



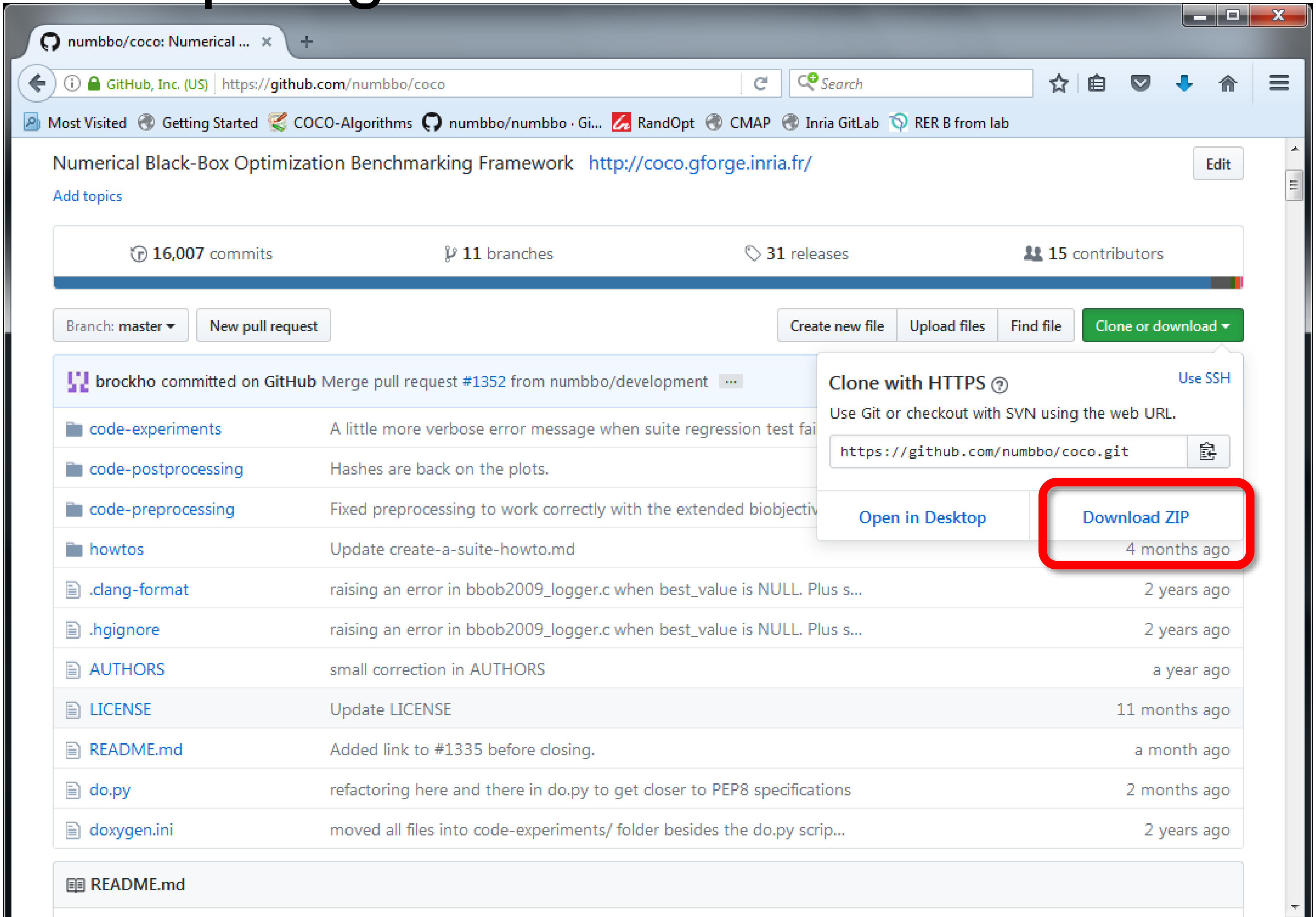
https://github.com/numbbo/coco



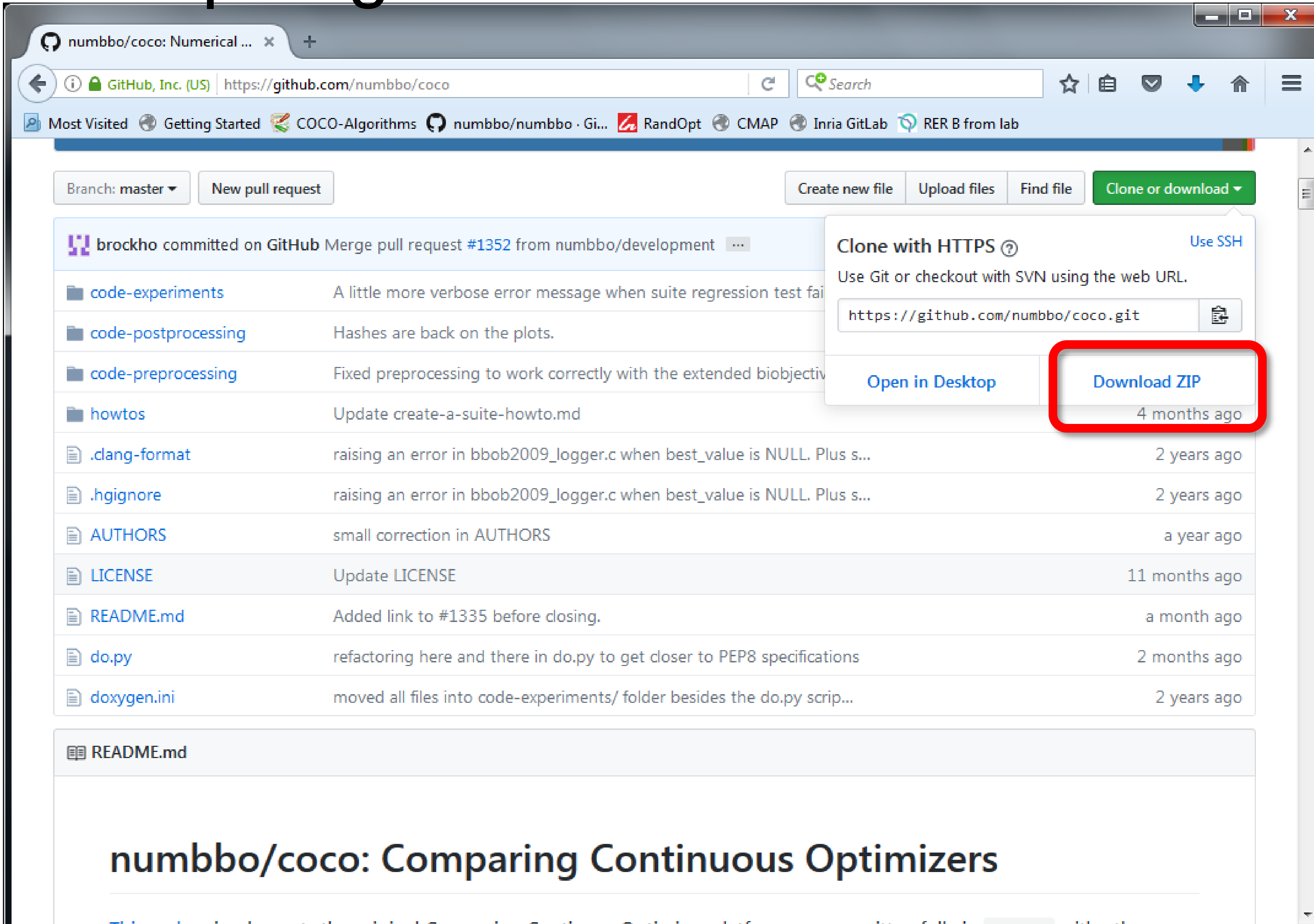
https://github.com/numbbo/coco



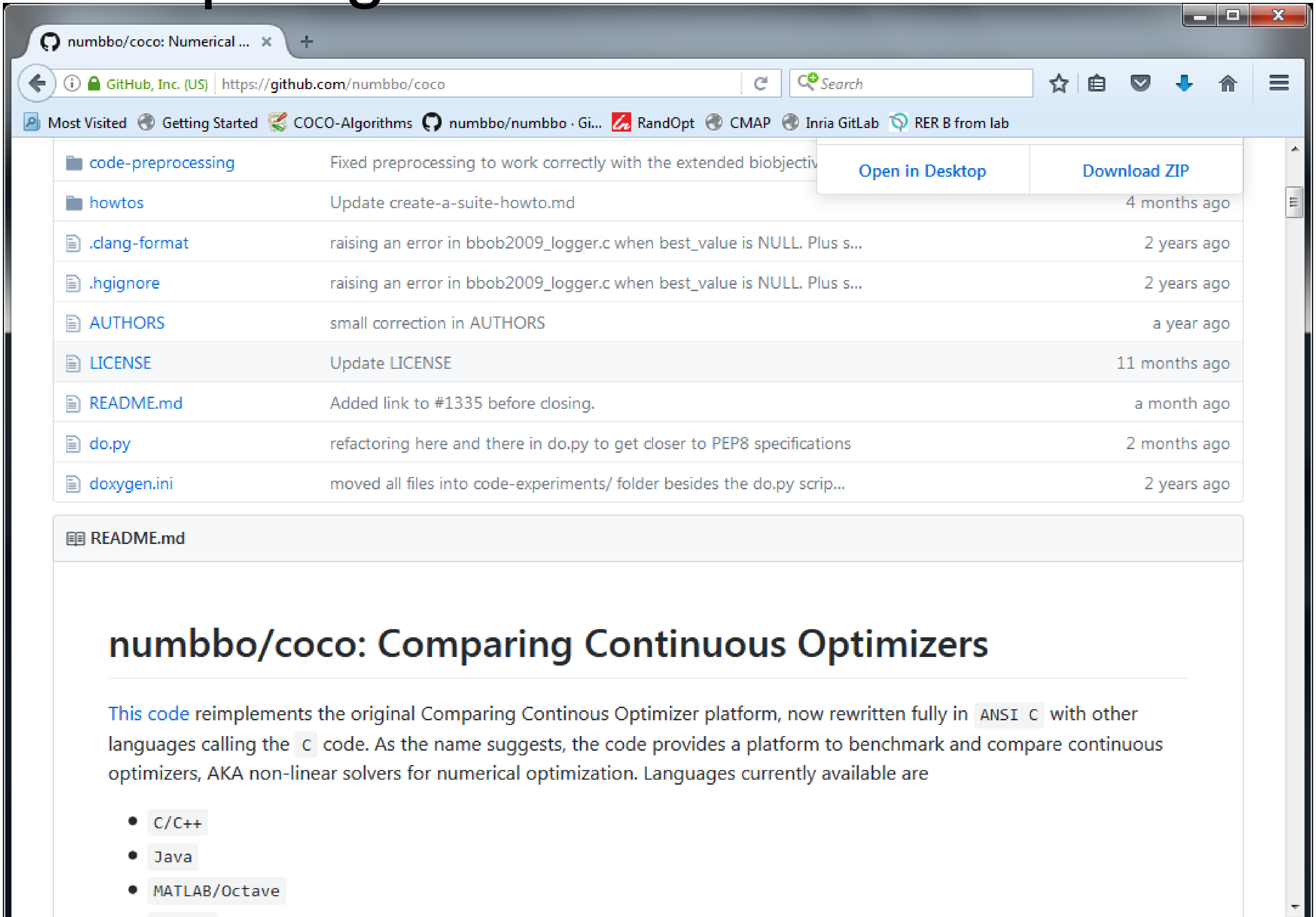
https://github.com/numbbo/coco



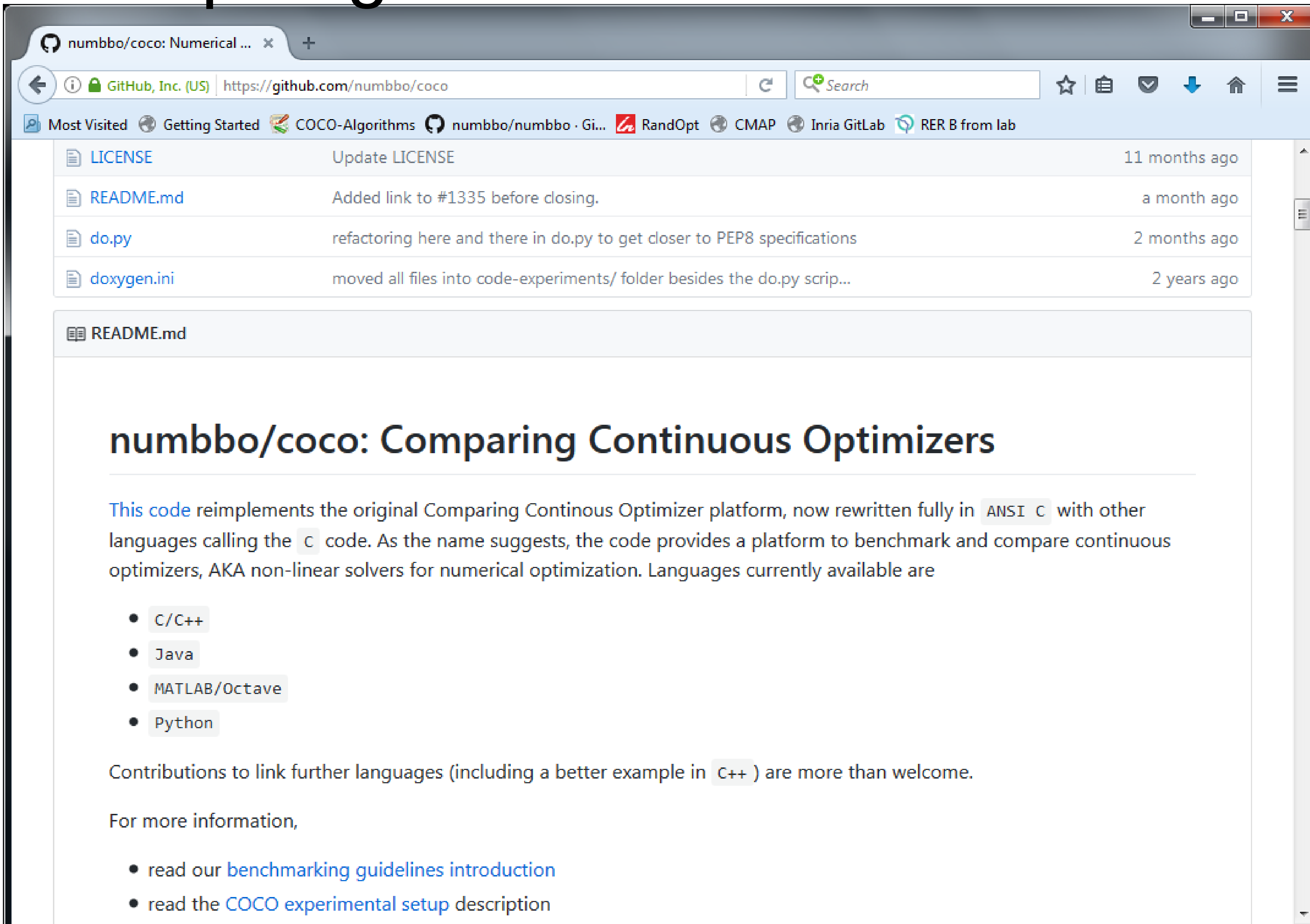
https://github.com/numbbo/coco



https://github.com/numbbo/coco



https://github.com/numbbo/coco



https://github.com/numbbo/coco

numbbo/coco: Numerical ...

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

Most Visited | Getting Started | COCO-Algorithms | numbbo/numbbo · Gi... | RandOpt | CMAP | Inria GitLab | RER B from lab

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

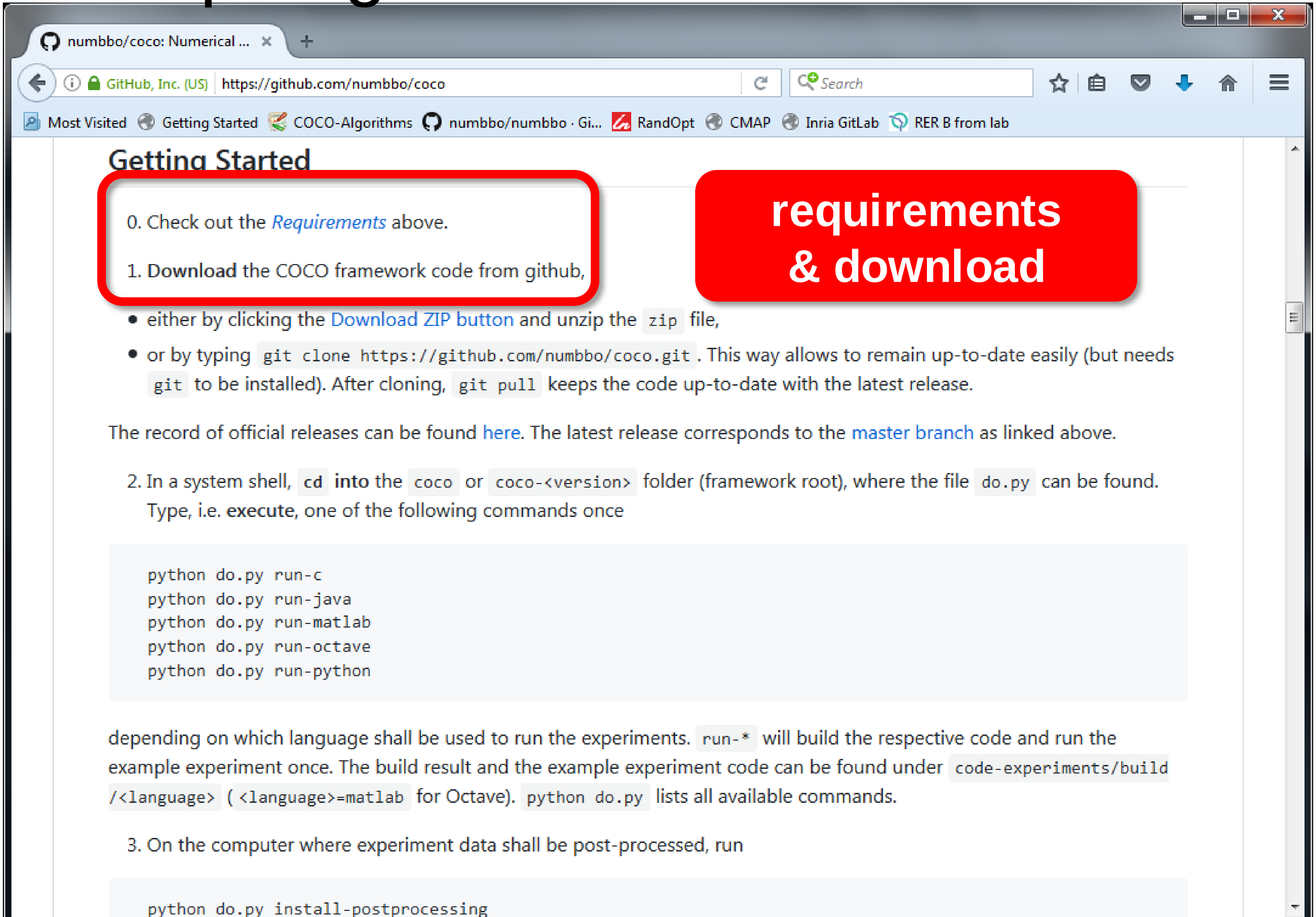
- `C/C++`
- `Java`
- `MATLAB/Octave`
- `Python`

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` and `bbob-biobj-ext` COCO multi-objective functions testbed documentation and the specificities of the performance assessment for the bi-objective testbeds.
- consult the BBOB workshops series,
- consider to register here for news,
- see the previous COCO home page here and
- see the links below to learn more about the ideas behind CoCO.

https://github.com/numbbo/coco



numbbo/coco: Numerical ...

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

Most Visited | Getting Started | COCO-Algorithms | numbbo/numbbo · Gi... | RandOpt | CMAP | Inria GitLab | RER B from lab

Getting Started

0. Check out the *Requirements* above.

1. **Download** the COCO framework code from github,

- either by clicking the Download ZIP button and unzip the `zip` file,
- or by typing `git clone https://github.com/numbbo/coco.git`. This way allows to remain up-to-date easily (but needs `git` to be installed). After cloning, `git pull` keeps the code up-to-date with the latest release.

The record of official releases can be found here. The latest release corresponds to the master branch as linked above.

2. 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.

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

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

requirements & download

https://github.com/numbbo/coco

numbbo/coco: Numerical ...

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

Most Visited | Getting Started | COCO-Algorithms | numbbo/numbbo · Gi... | RandOpt | CMAP | Inria GitLab | RER B from lab

Getting Started

0. Check out the *Requirements* above.

1. **Download** the COCO framework code from github,

- either by clicking the Download ZIP button and unzip the `zip` file,
- or by typing `git clone https://github.com/numbbo/coco.git`. This way allows to remain up-to-date easily (but needs `git` to be installed). After cloning, `git pull` keeps the code up-to-date with the latest release.

The record of official releases can be found here. The latest release corresponds to the master branch as linked above.

2. 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

installation I: experiments

```
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.

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

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

https://github.com/numbbo/coco

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.

3. On the computer where experiment data shall b...

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

installation II: postprocessing

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.

4. **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). 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
- `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.

5. Now you can **run** your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

https://github.com/numbbo/coco

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.

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

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

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.

4. **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). 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
- Python read me and example experiment

coupling algo + COCO

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.

5. Now you can **run** your favorite algorithm on the bbob suite (for single-objective algorithms) or on the bbob-biobj and bbob-biobj-ext suites (for multi-objective algorithms). Output is automatically generated in the specified data result_folder . By now, more suites might be available, see below.

Simplified Example Experiment in Python

```python
import cocoex
import scipy.optimize

### input
suite_name = "bbob"
output_folder = "scipy-optimize-fmin"
fmin = scipy.optimize.fmin

### prepare
suite = cocoex.Suite(suite_name, "", "")
observer = cocoex.Observer(suite_name,
                           "result_folder: " + output_folder)

### go
for problem in suite:  # this loop will take several minutes
    problem.observe_with(observer)  # generates the data for
                                    # cocopp post-processing
    fmin(problem, problem.initial_solution)
```

Note: the actual example_experiment.py contains more advanced things like restarts, batch experiments, other algorithms (e.g. CMA-ES), etc.

https://github.com/numbbo/coco

numbbo/coco at develop...

https://github.com/numbbo/coco/tree/development

Most Visited | Getting Started | COCO-Algorithms | numbbo/numbbo · Gi... | RandOpt | CMAP | Inria GitLab | RER B from lab

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

5. Now you can **run** your favorite algorithm on the `bbob` suite (for single-objective algorithms) or on the `bbob-biobj` and `bbob-biobj-ext` suites (for multi-objective algorithms). Output is automatically generated in the specified data `result_folder`. By now, more suites might be available, see below.

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

```
python -m cocopp [-o OUTPUT_FOLDERNAME] YOURDAT
```

running the experiment

Any subfolder in the folder arguments will be searched fo... n 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, p... file, usefu... the outpu...

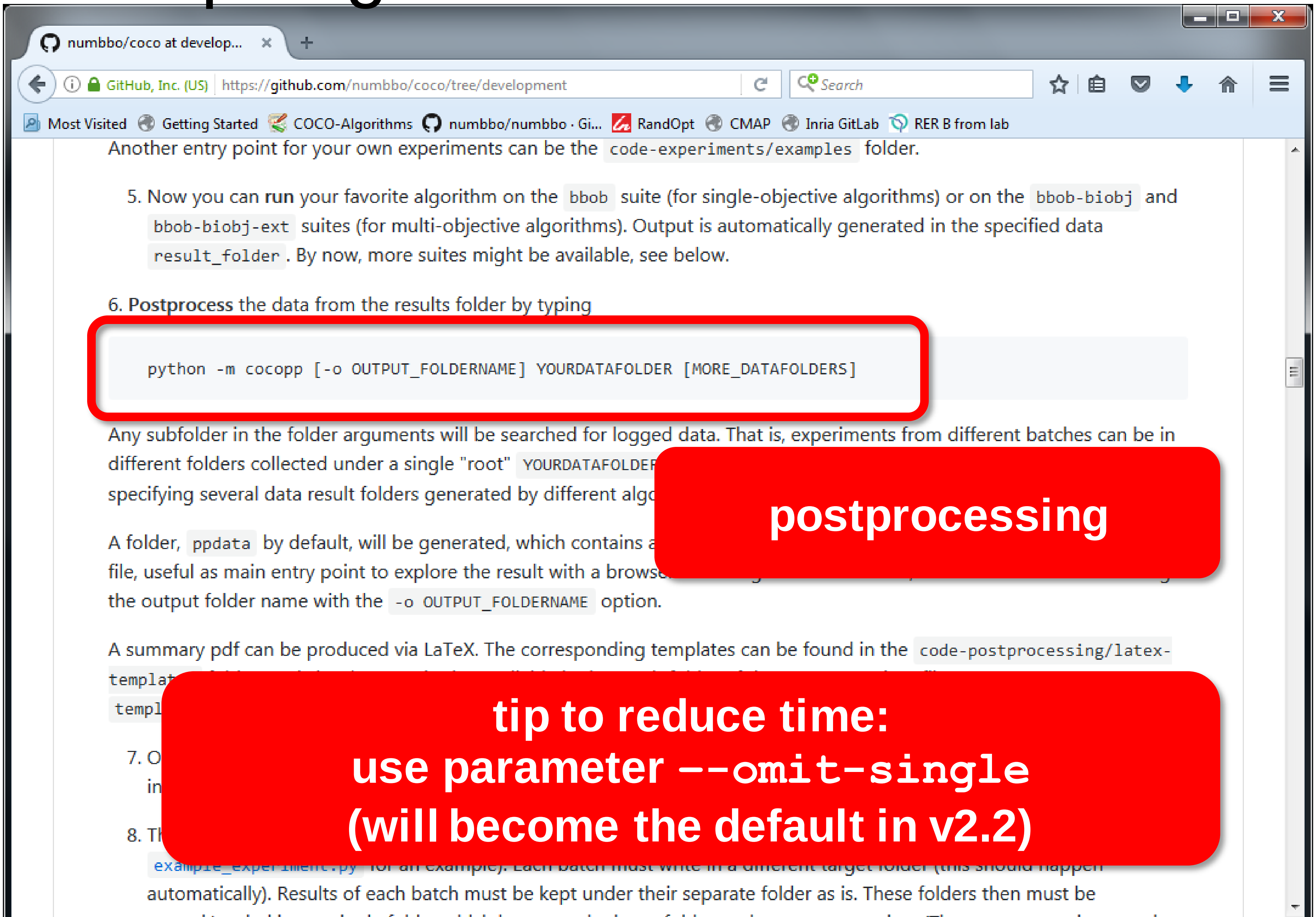
A summa... template... template...

7. Once... indep...

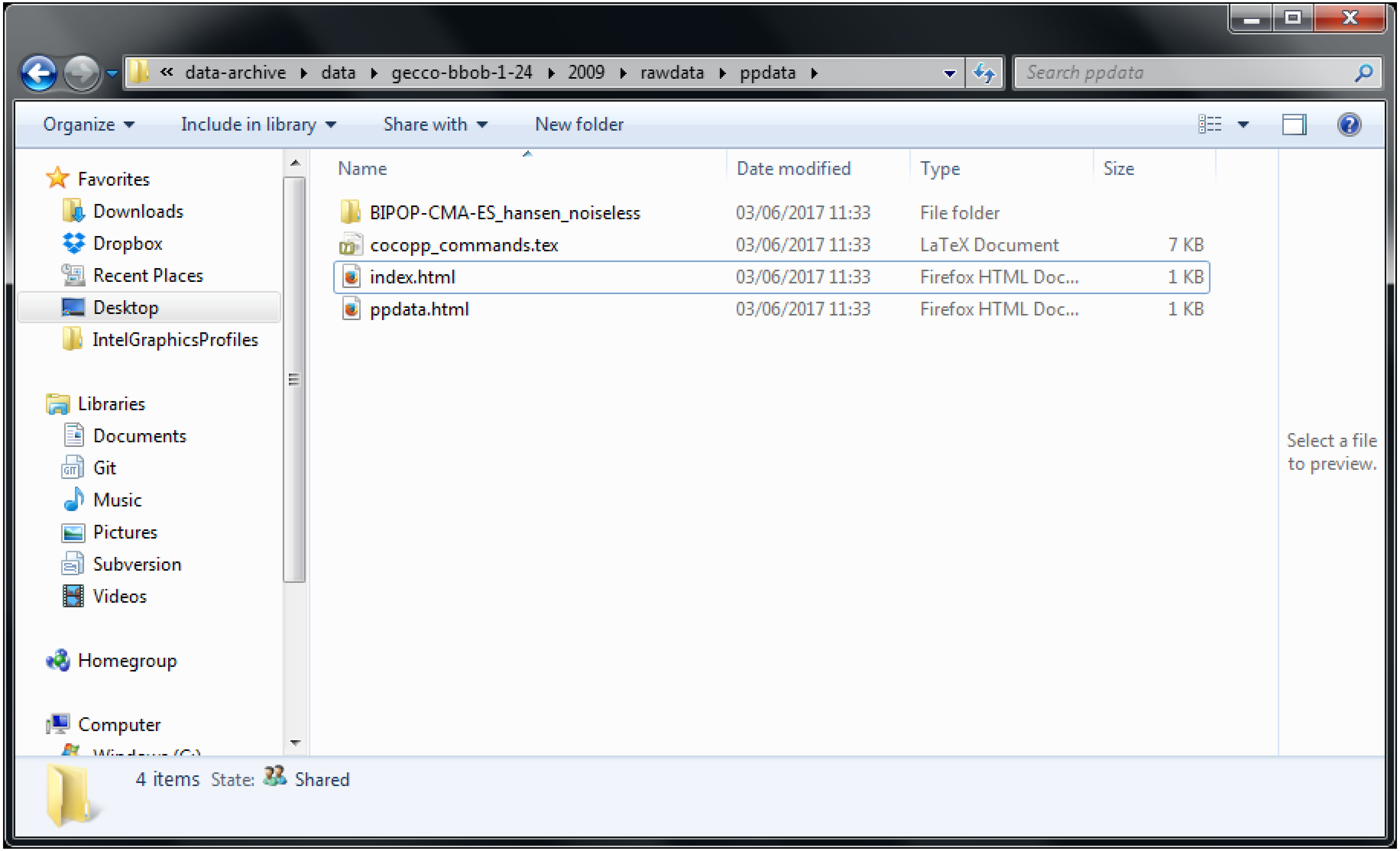
tip:
start with small #funevals (until bugs fixed ☺)
then increase budget to get a feeling
how long a "long run" will take

8. The experiments can be **parallelized** with any re-distribution of single problem instances to batches (see `example_experiment.py` for an example). Each batch must write in a different target folder (this should happen automatically). Results of each batch must be kept under their separate folder as is. These folders then must be

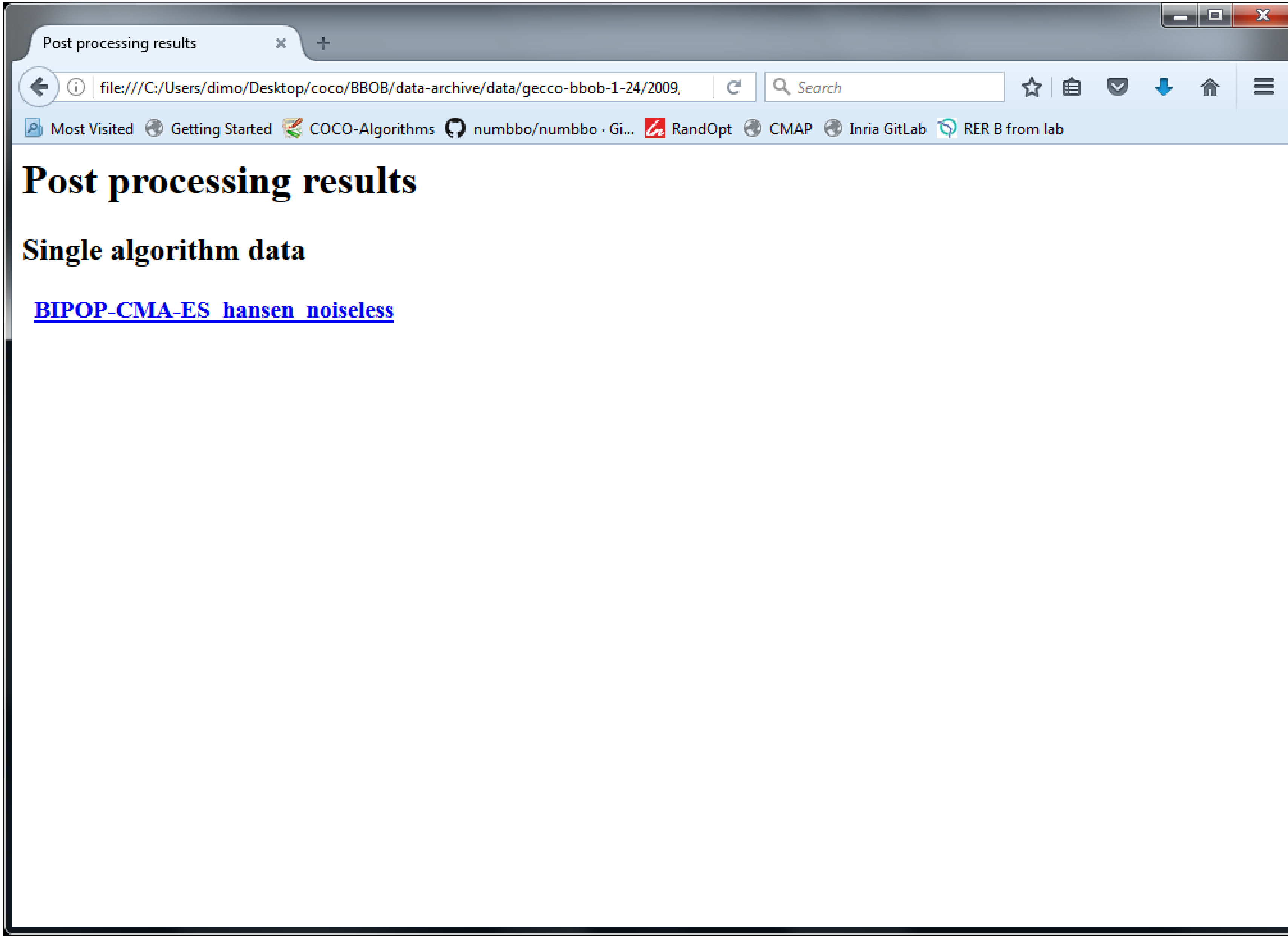
https://github.com/numbbo/coco



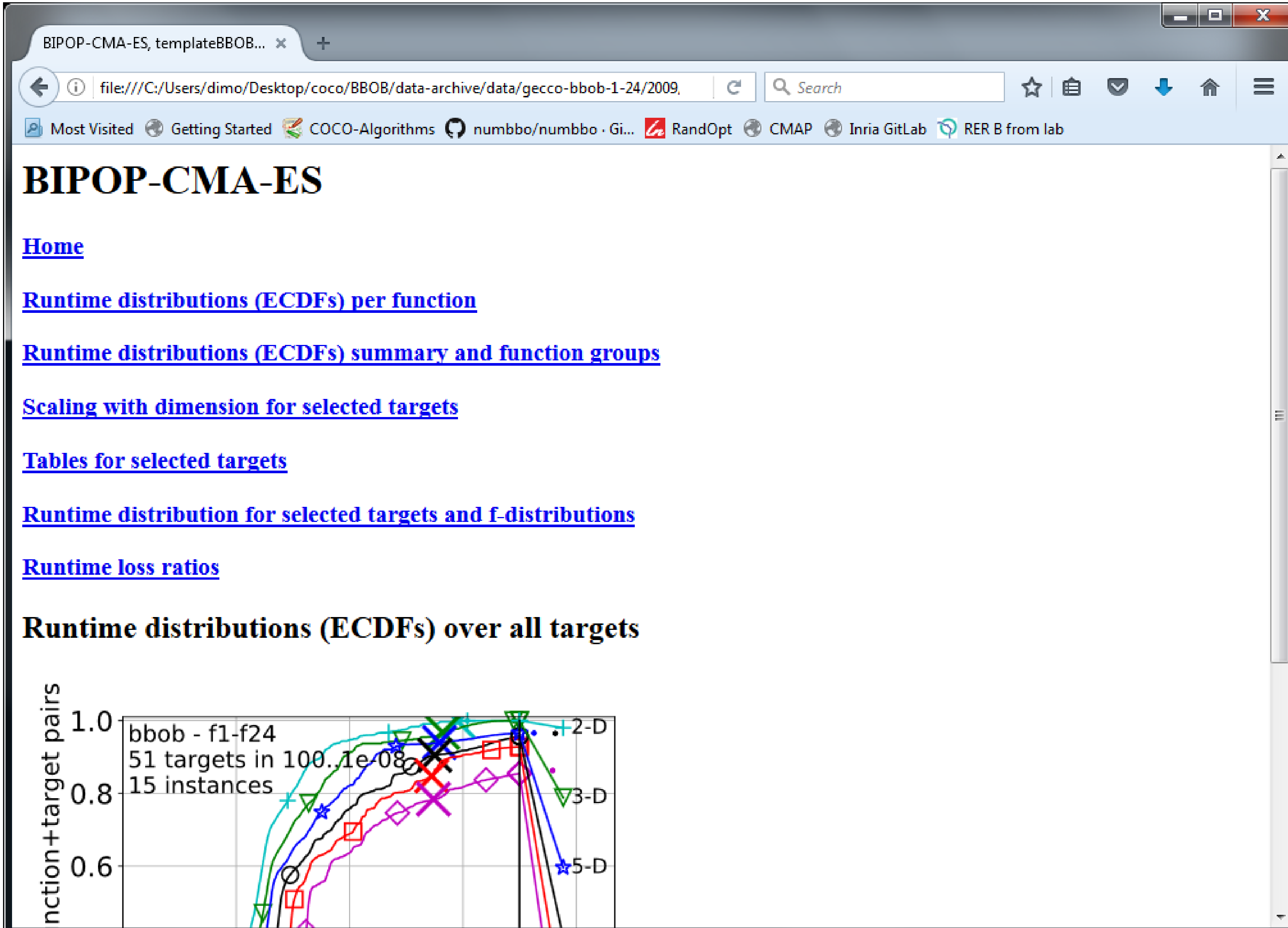
Result Folder



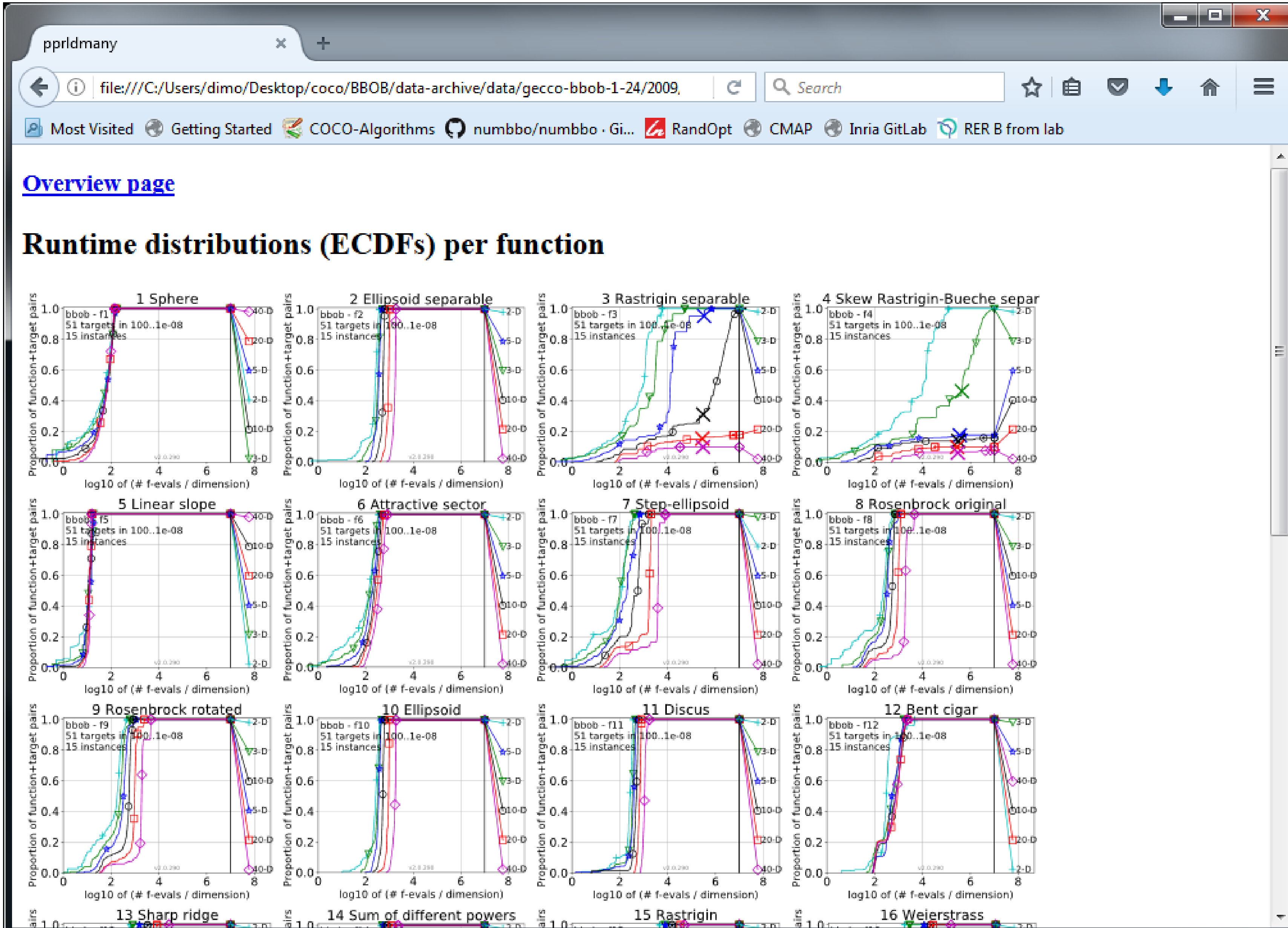
Automatically Generated Results



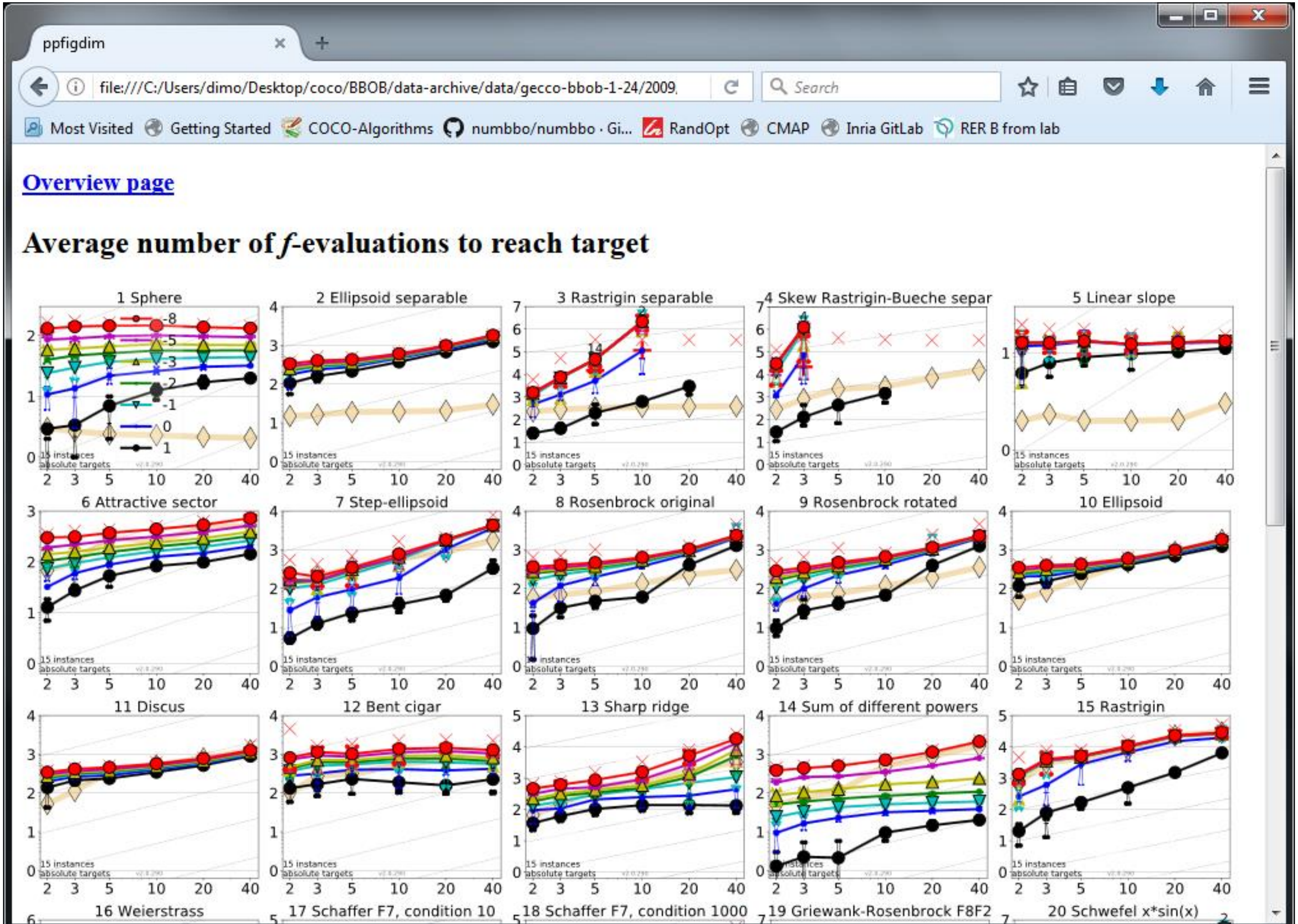
Automatically Generated Results



Automatically Generated Results



Automatically Generated Results



doesn't look too complicated, does it?

[the devil is in the details ☺]

Course Overview

1	Mon, 18.9.2017	today's lecture: more infos in the end
	Tue, 19.9.2017	groups defined via wiki
		everybody went (actively!) through the Getting Started part of github.com/numbbo/coco
2	Wed, 20.9.2017	lecture "Benchmarking", final adjustments of groups everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture)
3	Fri, 22.9.2017	lecture "Introduction to Continuous Optimization"
4	Fri, 29.9.2017	lecture "Gradient-Based Algorithms"
5	Fri, 6.10.2017	lecture "Stochastic Algorithms and DFO"
6	Fri, 13.10.2017	lecture "Discrete Optimization I: graphs, greedy algos, dyn. progr." deadline for submitting data sets
	Wed, 18.10.2017	deadline for paper submission
7	Fri, 20.10.2017	final lecture "Discrete Optimization II: dyn. progr., B&B, heuristics"
	Thu, 26.10.2017 / Fri, 27.10.2017	oral presentations (individual time slots)
	after 30.10.2017	vacation aka learning for the exams
	Fri, 10.11.2017	written exam

All deadlines:
23:59pm Paris time

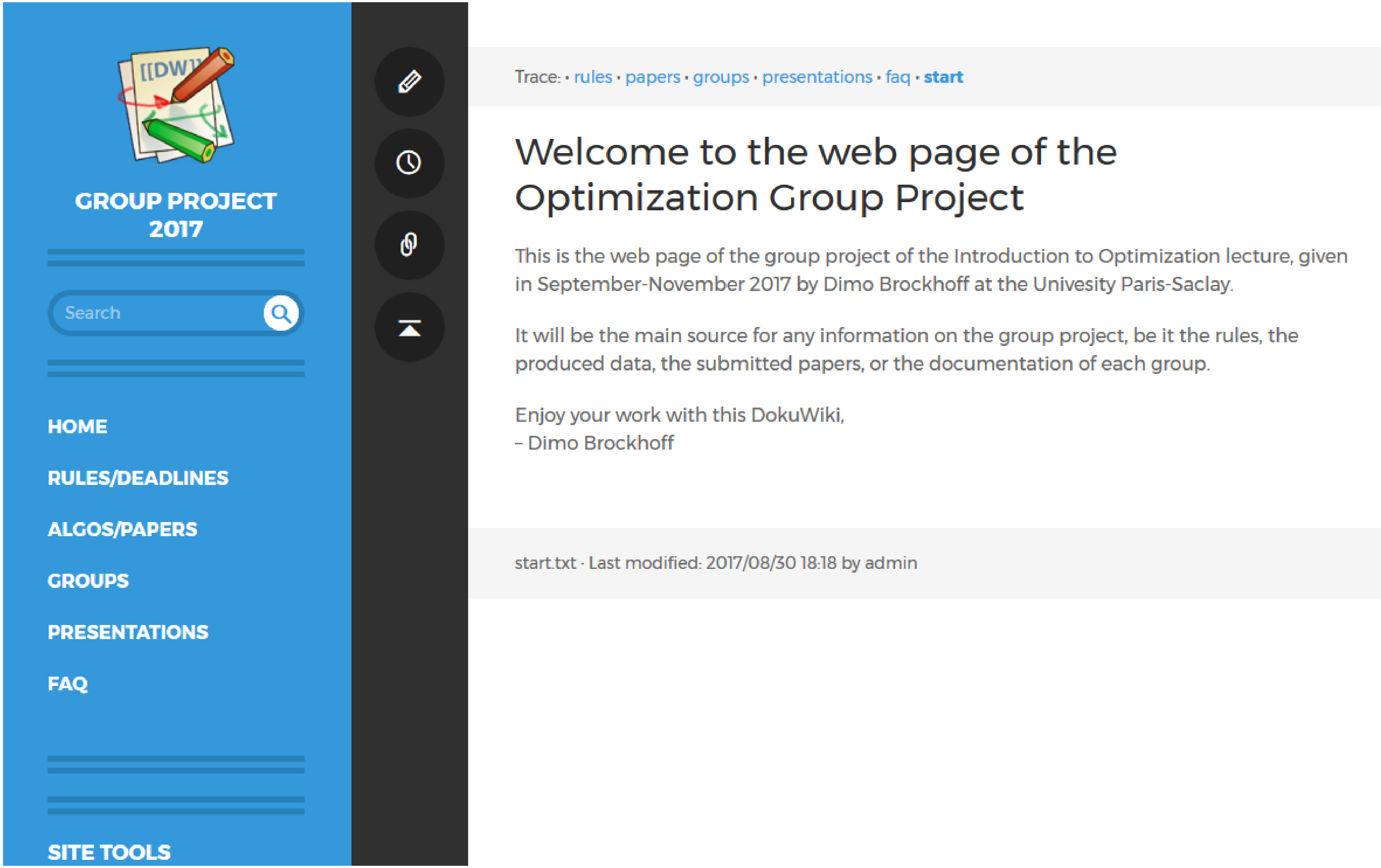
Group Project: Remark

both report and talk should be in English

[at the time being, THE scientific language]

Group Project Wiki

http://randopt.gforge.inria.fr/teaching/optimization-Saclay/groupproject2017/



Group Project Wiki

- to be found at
  - http://randopt.gforge.inria.fr/teaching/optimization-Saclay/groupproject2017/
  - 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
- and coordinate the assignments of all of you to groups with paper/algorithm and programming language (by tomorrow!)
  - 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

Group Project: Recommendations

- Do not start working last minute.

  Understanding an algorithm, implementing and testing it always takes time.

- Get an overview of what COCO is and does by reading the General Introduction to COCO and the documents on performance assessment with COCO to get an idea of how to read the main plots.

- Consider using a version control system for your code (and potentially for your final report and slides as well).

  Github/Gitlab might come in handy

- Test your software extensively. Optimally, write (unit) tests before the actual code.

- Again: run (very) short experiments first, then increase budget.

Course Overview

1	Mon, 18.9.2017	today's lecture: more infos in the end
	Tue, 19.9.2017	groups defined via wiki
		everybody went (actively!) through the Getting Started part of github.com/numbbo/coco
2	Wed, 20.9.2017	lecture "Benchmarking", final adjustments of groups everybody can run and postprocess the example experiment (~1h for final questions/help during the lecture)
3	Fri, 22.9.2017	lecture "Introduction to Continuous Optimization"
4	Fri, 29.9.2017	lecture "Gradient-Based Algorithms"
5	Fri, 6.10.2017	lecture "Stochastic Algorithms and DFO"
6	Fri, 13.10.2017	lecture "Discrete Optimization I: graphs, greedy algos, dyn. progr." deadline for submitting data sets
	Wed, 18.10.2017	deadline for paper submission
7	Fri, 20.10.2017	final lecture "Discrete Optimization II: dyn. progr., B&B, heuristics"
	Thu, 26.10.2017 / Fri, 27.10.2017	oral presentations (individual time slots)
	after 30.10.2017	vacation aka learning for the exams
	Fri, 10.11.2017	written exam

All deadlines:
23:59pm Paris time

Conclusions

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 tomorrow: build the groups and decide on an algorithm

by Wednesday:

- go through the "Getting Started" of COCO
- collect the things that don't work (concrete questions)