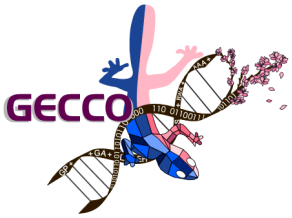



# A Short Introduction to COCO

Tea Tušar

Computational Intelligence Group
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia

July 15, 2018

Workshop on Game-Benchmark for Evolutionary Algorithms
Genetic and Evolutionary Computation Conference, GECCO 2018
Kyoto, Japan

# Why benchmark optimization algorithms?

No free lunch theorem $\Rightarrow$ No algorithm works best for all optimization problems

Purpose of benchmarking: To be able to select the best algorithm for the given real-world optimization problem

Preconditions

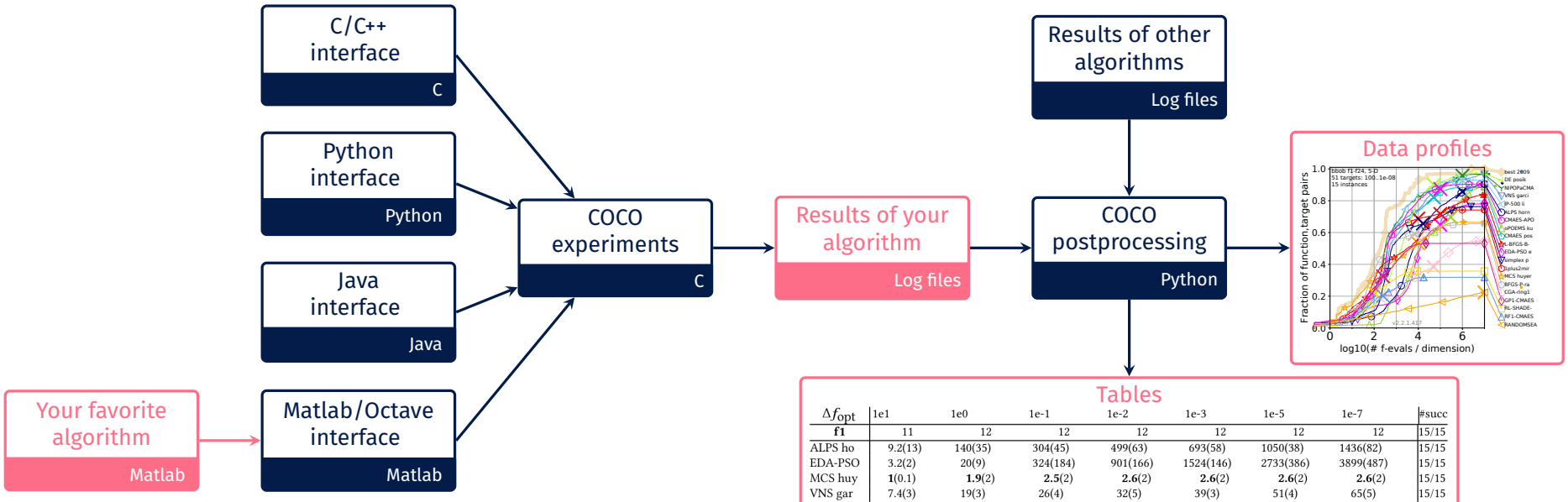
- The real-world problem with some known properties
- Test problems with similar properties to those of the real-world problem
- Results of several optimization algorithms on these test problems for any number of evaluations

# How to benchmark optimization algorithms?

## The COCO platform

- COCO (Comparing Continuous Optimizers)
- **`https://github.com/numbbo/coco`**
- Automatized benchmarking of optimization algorithms
  - Test problems with known properties
  - Data of previously run algorithms available for comparison
  - Provides interfaces to C/C++, Python, Java, Matlab/Octave
- Being developed at Inria Saclay, France, since 2007

# Benchmarking with COCO

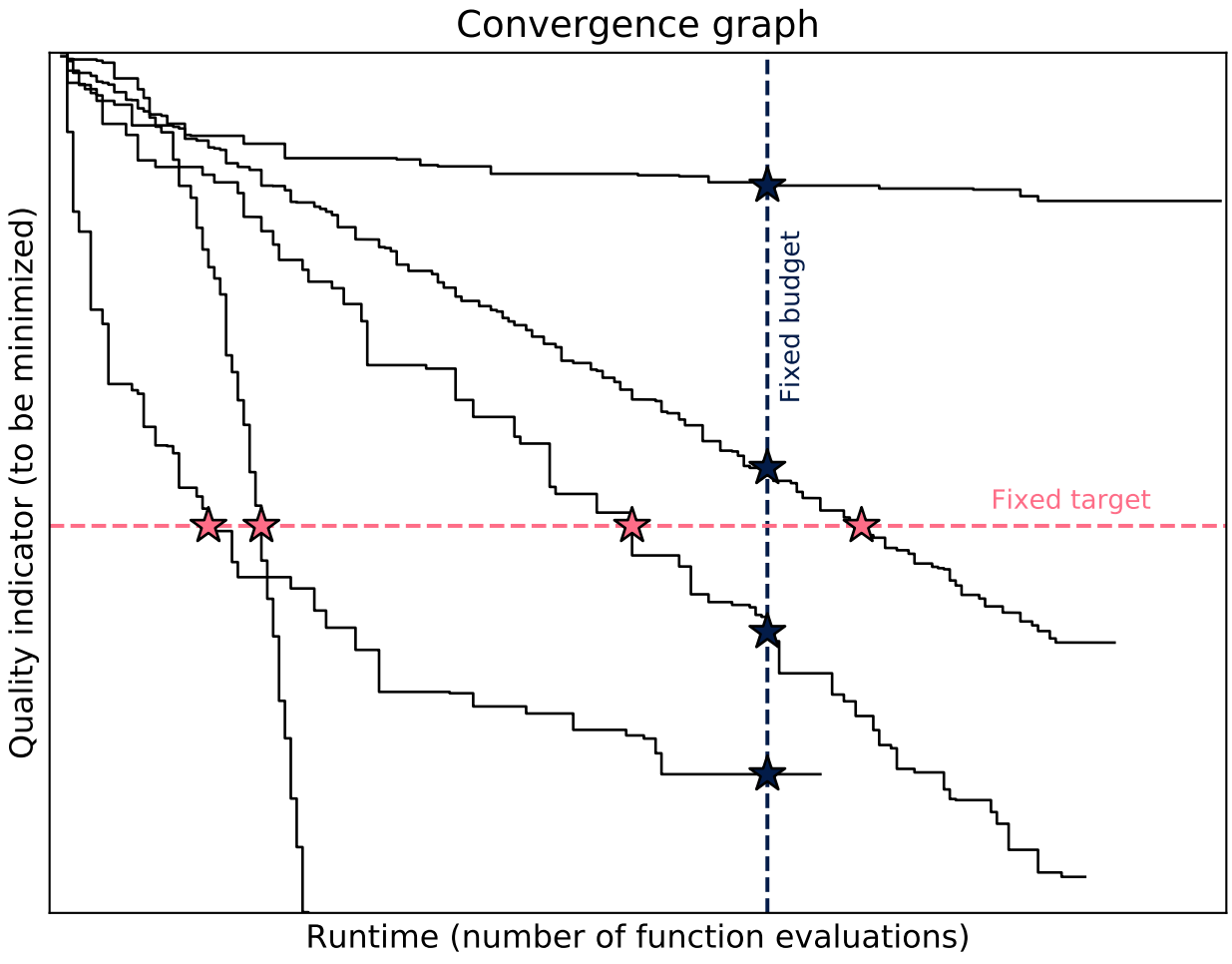



| $\Delta f_{\text{opt}}$ | 1e1 | 1e0 | 1e-1 | 1e-2 | 1e-3 | 1e-5 | 1e-7 | #succ |
|---|---|---|---|---|---|---|---|---|
| **f1** | 11 | 12 | 12 | 12 | 12 | 12 | 12 | 15/15 |
| ALPS ho | 9.2(13) | 140(35) | 304(45) | 499(63) | 693(58) | 1050(38) | 1436(82) | 15/15 |
| EDA-PSO | 3.2(2) | 20(9) | 324(184) | 901(166) | 1524(146) | 2733(386) | 3899(487) | 15/15 |
| MCS huy | **1**(0.1) | **1.9**(2) | **2.5**(2) | **2.6**(2) | **2.6**(2) | **2.6**(2) | **2.6**(2) | 15/15 |
| VNS gar | 7.4(3) | 19(3) | 26(4) | 32(5) | 39(3) | 51(4) | 65(5) | 15/15 |
| 1plus2m | **1.5**(1.0) | 5.0(1) | 8.1(2) | 12(1) | 15(3) | 22(3) | 28(3) | 15/15 |
| oPOEMS | 102(45) | 139(15) | 334(147) | 779(201) | 1215(164) | 2034(352) | 2843(178) | 15/15 |
| CMAES p | **2.3**(1) | 8.7(3) | 15(6) | 22(5) | 28(4) | 41(4) | 53(6) | 15/15 |
| DE posi | 5.0(4) | 21(9) | 39(9) | 61(11) | 83(7) | 124(9) | 167(15) | 15/15 |
| NIPOPaC | **2.7**(2) | 7.5(3) | 14(2) | 20(4) | 26(3) | 37(3) | 51(5) | 15/15 |
| CGA-rin | 6.7(7) | 28(10) | 67(13) | 154(46) | 271(88) | 2085(1031) | 4.7e4(4e4) | 0/15 |
| IP-500 | 3.0(1.0) | 7.6(1) | 13(3) | 21(2) | 27(3) | 41(7) | 54(6) | 15/15 |
| simplex | 21(0.7) | 64(28) | 77(30) | 84(18) | 87(15) | 92(18) | 96(12) | 15/15 |
| L-BFGS- | **1.4**(0.0) | **1.3**(0.0) | **1.3**(0.0) | **1.4**(0.3) | **1.4**(0.3) | **1.4**(0.2) | **1.4**(0.3) | 15/15 |
| RL-SHAD | **2.4**(3) | 9.3(4) | 14(5) | 20(3) | 26(7) | 41(24) | 205(323) | 1/15 |
| GP1-CMA | **2.3**(2) | 6.1(1) | 9.2(2) | 13(3) | 16(4) | 21(5) | 31(5) | 15/15 |
| RF1-CMA | **2.6**(1) | 8.3(2) | 15(2) | 25(4) | 44(16) | 217(245) | 758(1022) | 1/15 |
| RANDOMS | 3.2(4) | 1037(4718) | 1.9e5(9e4) | 2.7e7(4e7) | ∞ | ∞ | ∞ *5e7* | 0/15 |
| CMAES-A | **2.3**(3) | 6.0(3) | 14(3) | 19(3) | 26(1) | 39(5) | 52(4) | 15/15 |
| BFGS-P- | **1.5**(0) | **1.3**(0) | **1.3**(0) | **1.3**(0) | **1.3**(0) | **1.3**(0) | **1.8**(0.6) | 15/15 |

Requirements: C compiler and Python (other languages are optional)

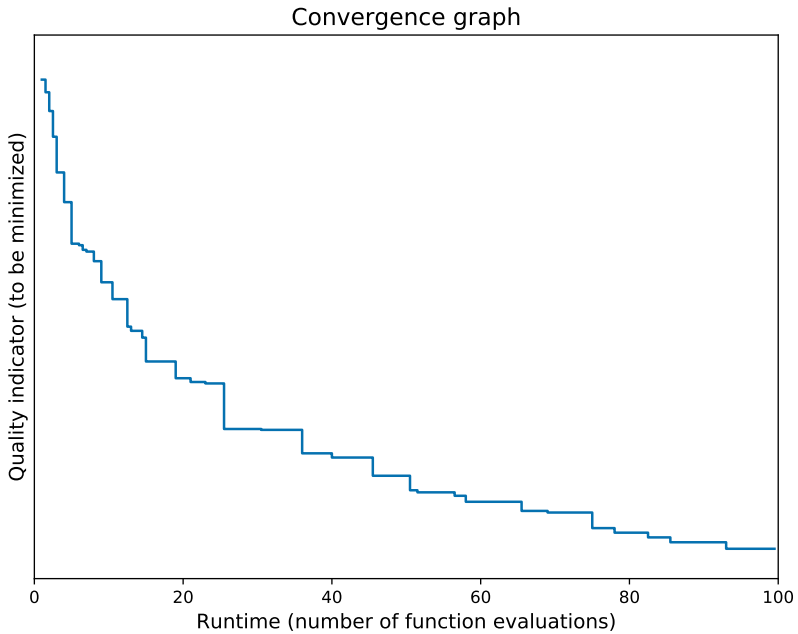
# The fixed-target approach

Interested in the runtime (number of function evaluations) needed to achieve a target value



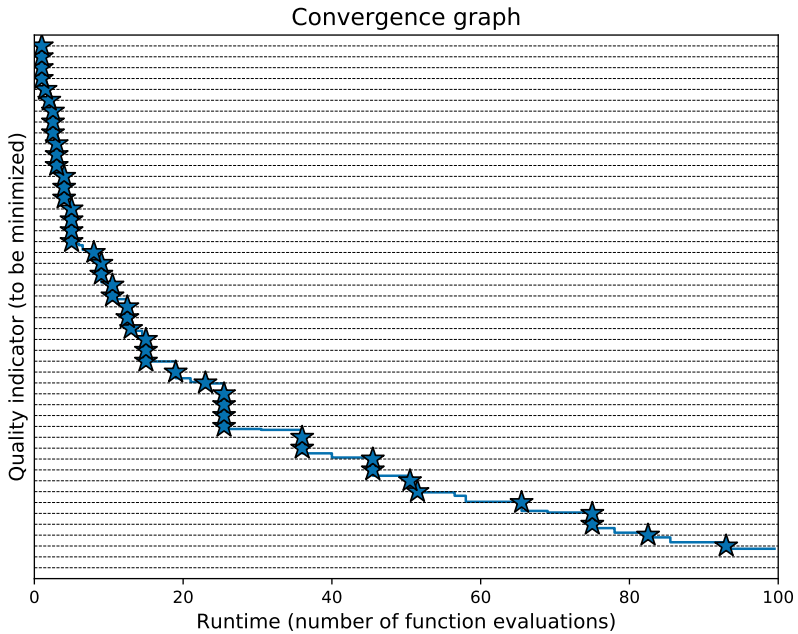
# Data profile

The data profile is the empirical cumulative distribution function (ECDF) of the recorded runtimes



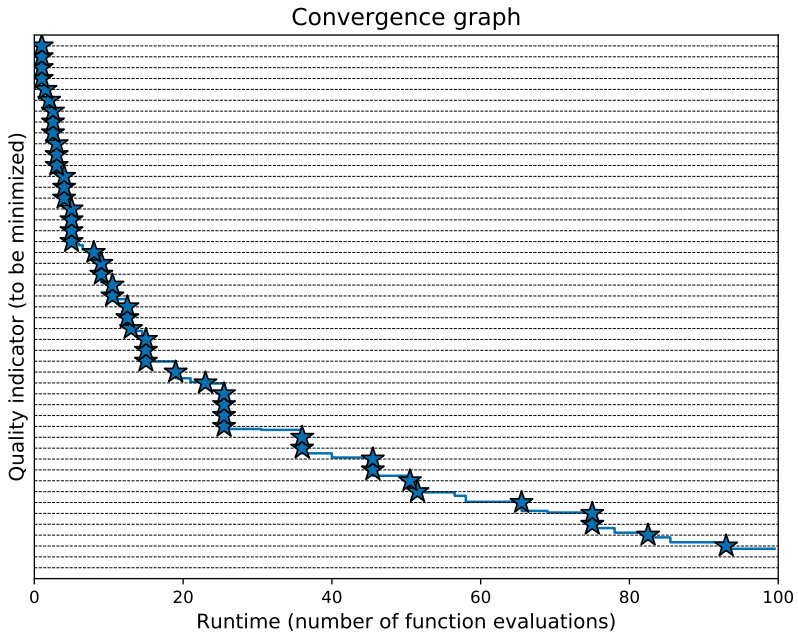
# Data profile

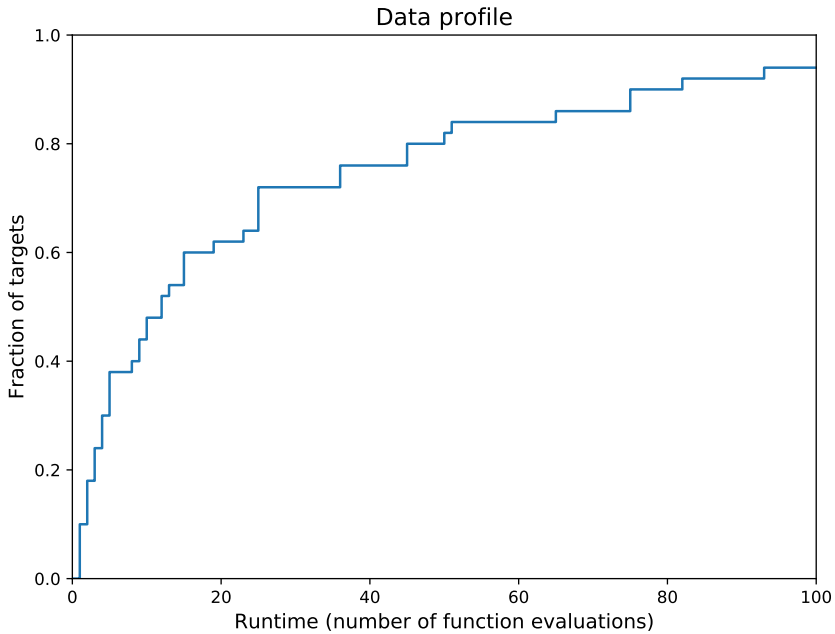
The data profile is the empirical cumulative distribution function (ECDF) of the recorded runtimes



# Data profile

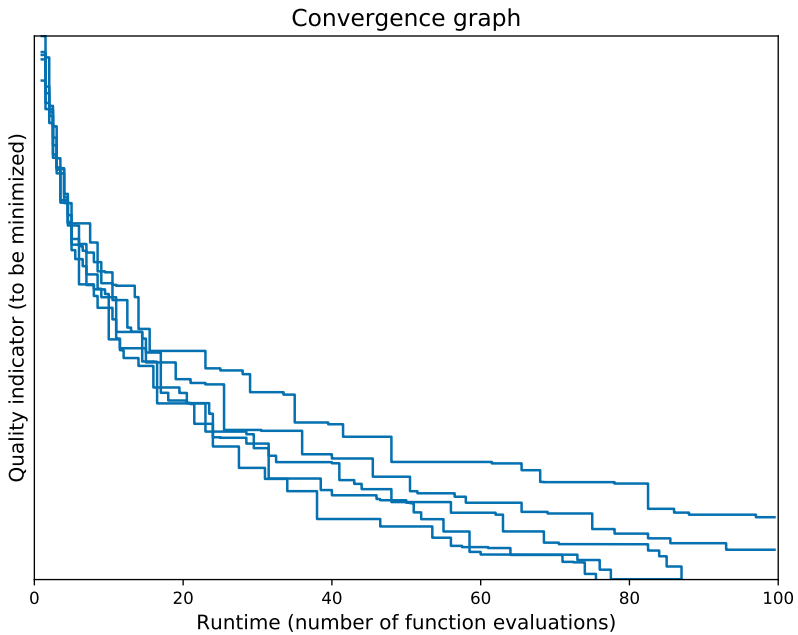
The data profile is the empirical cumulative distribution function (ECDF) of the recorded runtimes



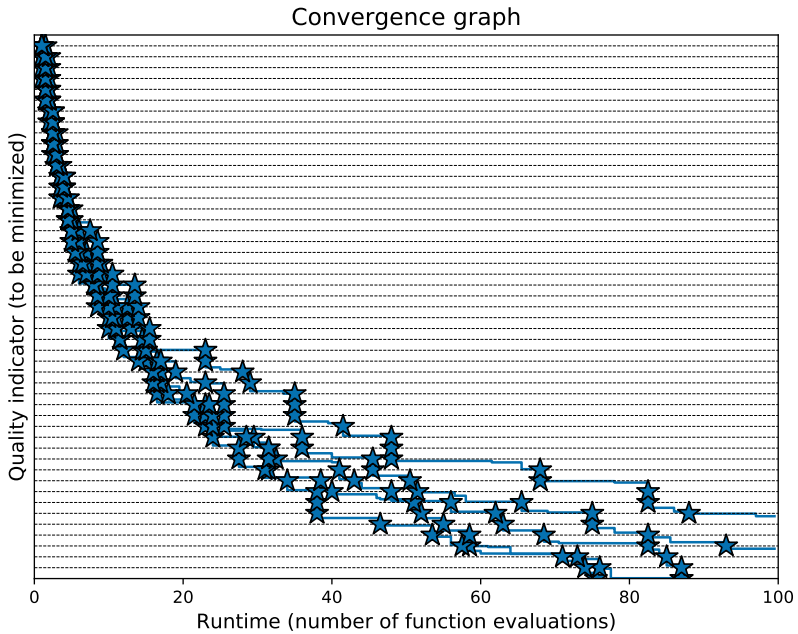
# Data profile

## Data profiles can aggregate performance over multiple runs



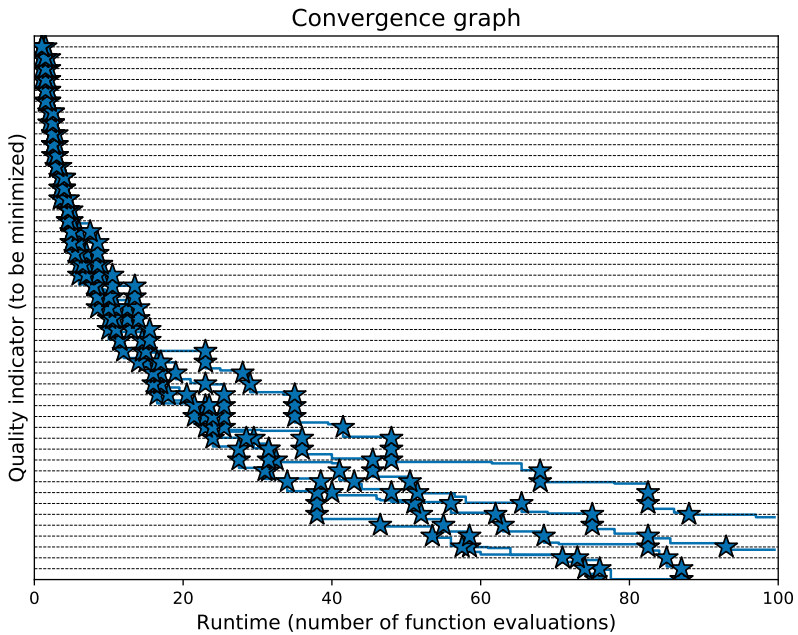
# Data profile

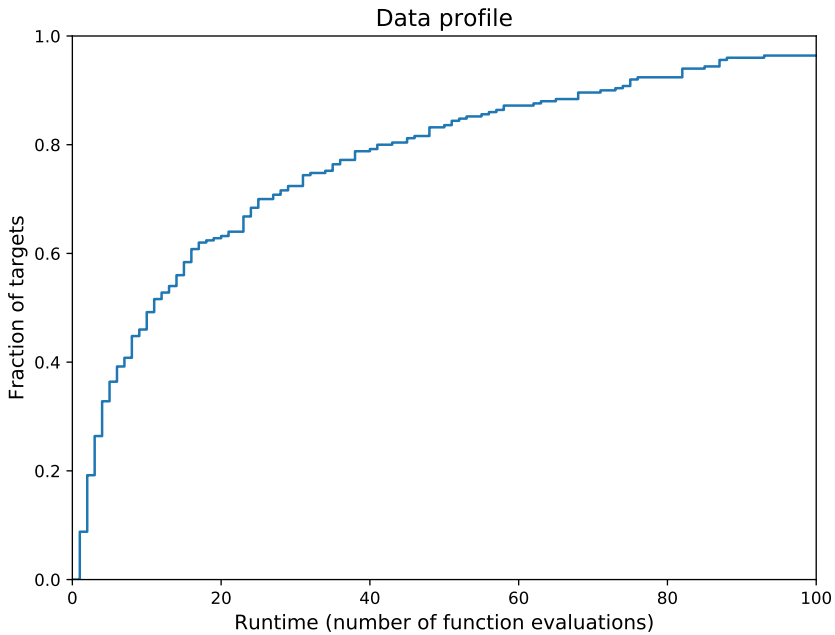
## Data profiles can aggregate performance over multiple runs



# Data profile

## Data profiles can aggregate performance over multiple runs



# COCO test suites

## Test suites and algorithm results

- **bbob** test suite with 24 functions (173 algorithms)
- **bbob-noisy** test suite with 30 functions (45 algorithms)
- **bbob-biobj** test suite with 55 functions (16 algorithms)

Algorithm results collected at 9 BBOB Workshops (since 2009, mostly at GECCO conferences)

## Under development

- Suite with constrained problems
- Suite with large-scale problems
- Suites with real-world problems