

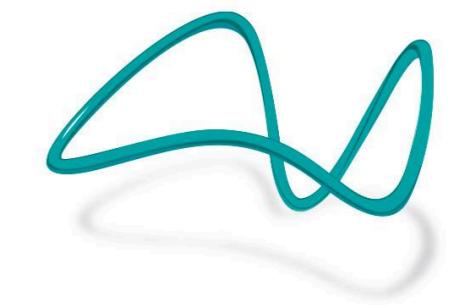
第四届广州深圳非光滑优化研讨会

2026年3月6日-3月8日



OptiProfiler: A Benchmarking Platform for Optimization Solvers

Cunxin Huang, Tom M. Ragonneau, Zaikun Zhang
(corresponding: cun-xin.huang@connect.polyu.hk)



Contribution highlights



We present **OptiProfiler**, a benchmarking platform for optimization solvers with a current focus on **derivative-free optimization (DFO)**. OptiProfiler provides:

- **Automated benchmarking** with performance profiles, data profiles, and log-ratio profiles,
- **Flexible testing environments** with multiple built-in features and problem libraries,
- **Easy-to-use interface** requiring only a few lines of code.

Website & Documentation

<https://optprof.com>



Why OptiProfiler?

Benchmarking is essential in optimization research, yet existing tools have limitations:

- Performance profiles are often implemented with basic code aimed at experts,
- Many platforms lack support for customizable testing environments,
- Reproducibility and result tracking are often overlooked.

OptiProfiler addresses these needs:



Simple usage for beginners



Multiple degrees of freedom for experts



Automatic generation of high-quality profiles



Reliable methodology for benchmarking



Trackable experimental results for reproducibility

Quick start

Installation:

```
git clone https://github.com/optiprofiler/  
optiprofiler.git  
addpath(genpath('optiprofiler/matlab'))
```

Simple usage:

```
benchmark({@fminsearch, @fminunc}) %  
Default  
benchmark({@solver1, @solver2}, 'noisy') % With  
feature  
  
options.feature_name = 'noisy';  
options.noise_level = 0.1;  
[scores, data] = benchmark(solvers, options)
```

Built-in features

OptiProfiler provides multiple testing environments through **features**:

Feature	Description
plain	Original problem
perturbed_x0	Randomly perturb initial guess
noisy	Add noise to function values
truncated	Truncate to given precision
permuted	Randomly permute variables
linearly_transformed	Apply linear transformation to the variables
random_nan	Randomly set values to NaN
quantized	Quantize function values

Example with customized noisy feature:

```
options.feature_name = 'noisy';  
options.noise_type = 'relative';  
options.distribution = 'gaussian';  
options.noise_level = 0.01;  
benchmark(solvers, options)
```

Problem library

Default: **S2MPJ** (by Gratton & Toint) — Problems in MATLAB, Python, Julia; no MEX files needed.

Built-in: **MatCUTEst** for MATLAB (by Zhang) and **PyCUTEst** for Python (by Fowkes & Roberts)

Problem selection:

```
options.ptype = 'u'; % Unconstrained  
options.mindim = 6;  
options.maxdim = 100;  
options.plibs = {'s2mpj', 'matcutes'};
```

Types: Unconstrained ('u'), Bound ('b'), Linear ('l'), Nonlinear ('n')

Benchmarking tools

Performance profile (Dolan & Moré, 2002; Moré & Wild, 2009):

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} : r_{p,s} \leq \alpha\}|,$$

where $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$ is the relative cost.

Data profile (Moré & Wild, 2009):

$$\delta_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\} \right|.$$

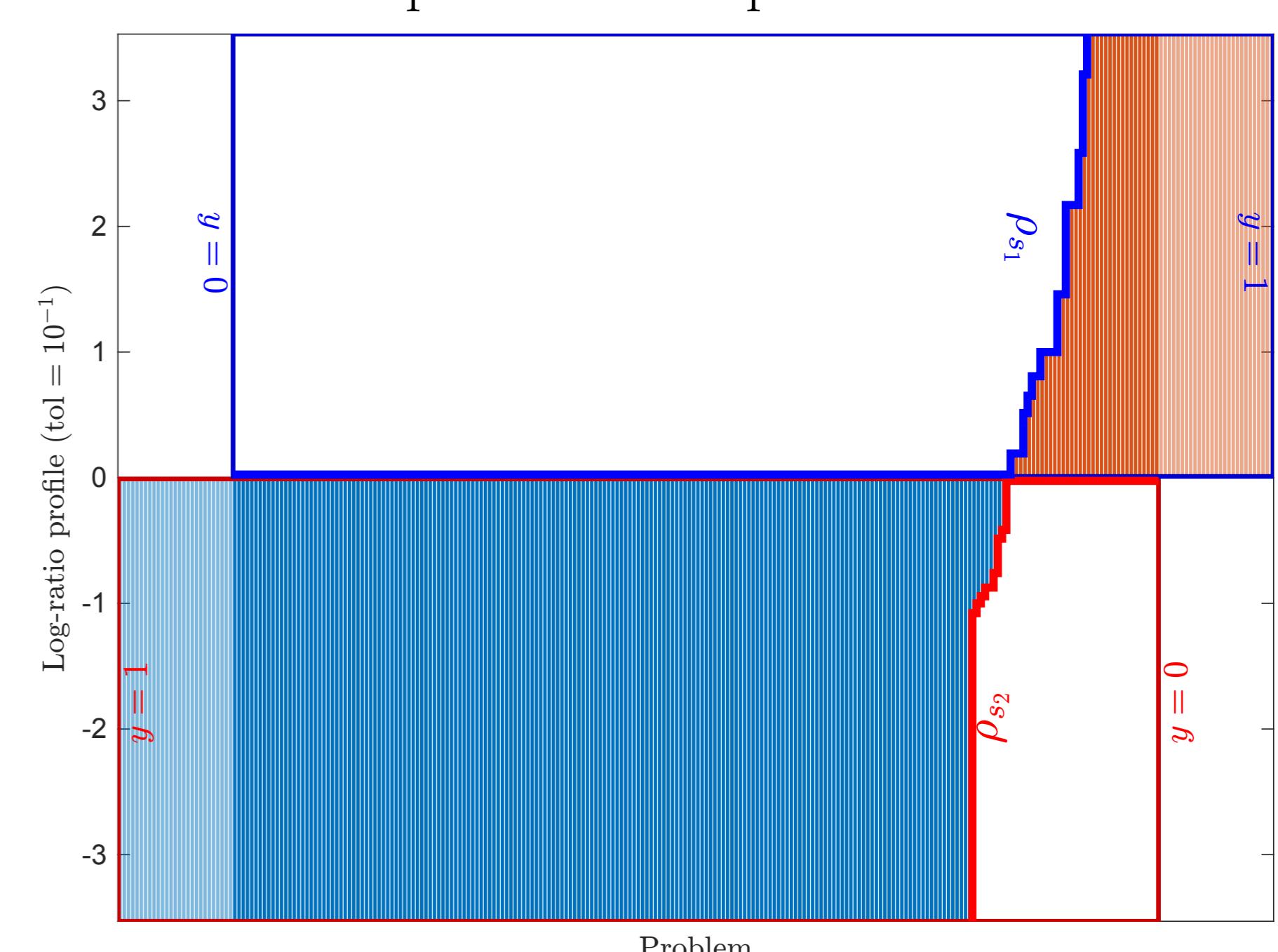
Log-ratio profile: When comparing two solvers, displays $\log_2(t_{p,s_1}/t_{p,s_2})$ sorted in ascending order.

History-based vs. Output-based:

- **History-based:** Cost to first reach convergence test
- **Output-based:** Total cost if output passes test

Equivalence: performance \leftrightarrow log-ratio

When comparing two solvers, the **shaded area** in log-ratio profile is related to the **Area Above the Curve** of performance profile.



Generated outputs

OptiProfiler automatically generates:

- Summary PDF of all profiles,
- Detailed profiles at multiple tolerance levels,
- Solver scores based on AUC,
- Time-stamped result folders.

