SIAM OP26 – EDINBURGH

OptiProfiler

Toward a DFO Benchmarking Ecosystem

Cunxin Huang

Joint work with Tom M. Ragonneau and Zaikun Zhang

The Hong Kong Polytechnic University

What Is a Benchmark?

CONCEPT

A benchmark turns comparison into a specified game.

Benchmark = Objects + Tasks + Scoring

Who to test?
DFO solvers
methods being compared

What to do?
Optimization problems
same suite, same budgets

How to quantify?
Different profiles
scores, logs, evidence

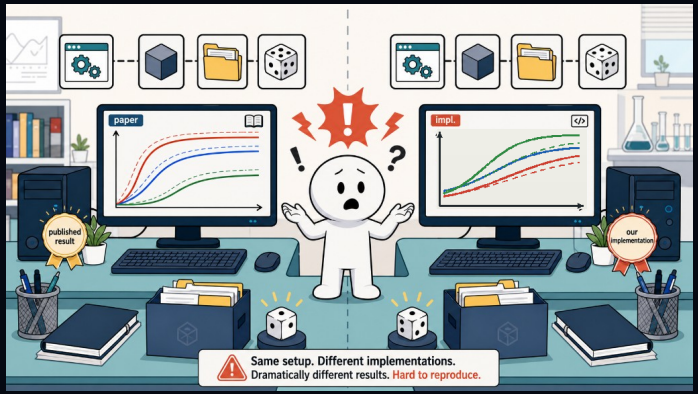
Experience I: The Same Benchmark Did Not Give the Same Story

PERSONAL EXPERIENCE

Solvers, or scripts?

1. **Reproduce**
   same paper, same solvers
2. **Replot**
   small implementation choices
3. **Disagree**
   curves move enough to matter

Need a trusted engine.



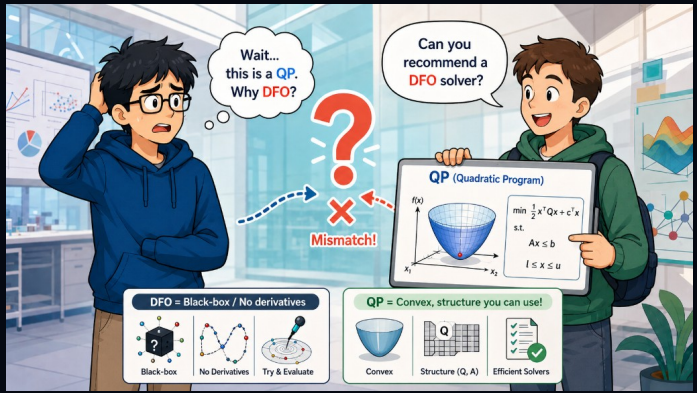
Experience II: Before Benchmarking, Ask the Right Question

PERSONAL EXPERIENCE

Use DFO only when it fits.

1. **User asks**
   "Which DFO solver?"
2. **We inspect**
   convex quadratic structure
3. **We redirect**
   use the structure first

Guide before the run.



FROM EXPERIENCES TO REQUIREMENT

Benchmarking DFO is not just running solvers; it is producing evidence people can reuse.

We need bench infra.

Not one script.

Not one solver run.

Reusable comparison pipeline.

GOOD INFRASTRUCTURE IS SMART

01	Simple	quick start, examples
02	Multiple	suites, features, budgets
03	Automatic	profiles, plots, reports
04	Reliable	accepted metrics and scores
05	Trackable	seeds, logs, run metadata

Decisions From Function Values Alone

SETTING

With no derivatives to inspect, empirical evidence carries much of the comparison.

Black-box problem

$$\min_{x \in \Omega} f(x)$$

Oracle evaluate $x \mapsto f(x)$

No gradients unavailable / costly

Expensive black-box models.

Standards

Problem libraries	CUTEst, S2MPJ, PyCUTEst, MatCUTEst
Profile views	performance, data, log-ratio
Solver families	Powell's solvers, ...

Profiles Generated for One Benchmark

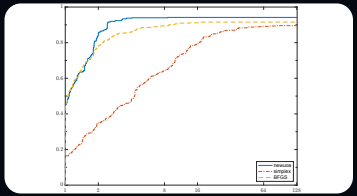
SHARED EVIDENCE LANGUAGE

One cost matrix $T = (t_{p,s})$; three standard views (plain experiment below).

Performance profile

$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\left|\{p : r_{p,s} \le \alpha\}\right|$

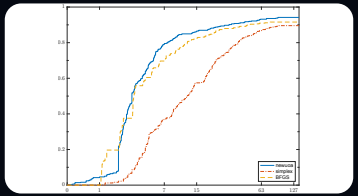
$r_{p,s} = t_{p,s} / \min_s t_{p,s}$



Data profile

$\delta_s(\alpha) = \frac{1}{|\mathcal{P}|}\left|\{p : c_{p,s} \le \alpha\}\right|$

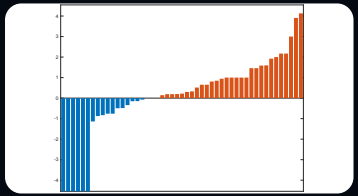
$c_{p,s} = t_{p,s} / (n_p + 1)$



Log-ratio profile

$\lambda_k = \mathrm{sort}(\{\log_2(t_{p,s_1}/t_{p,s_2})\})$

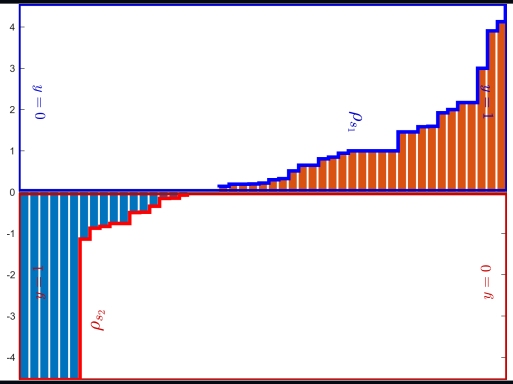
two solvers; one bar per problem



Literature: Dolan–Moré (2002); Moré–Wild (2009); Morales (2002); Shi et al. (2023).

Interesting Fact: Two Views, Same Comparison

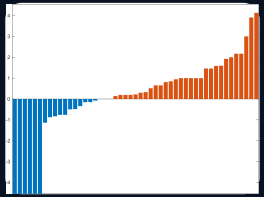
The log-ratio profile and the performance profile (semilogx) are **mutually inverse**.



Same Data: Split Halves, Recover Profiles

TRANSFORMATION (PLAIN EXPERIMENT)

Start from the log-ratio view; split, rotate, and recover standard performance profiles.

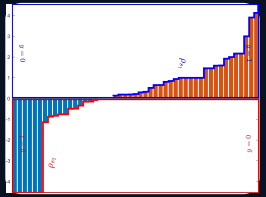


log-ratio profile

Same Data: Split Halves, Recover Profiles

TRANSFORMATION (PLAIN EXPERIMENT)

Start from the log-ratio view; split, rotate, and recover standard performance profiles.

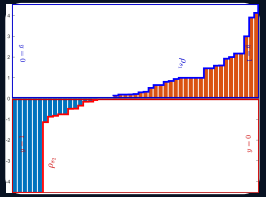


highlighted view

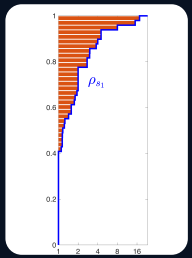
Same Data: Split Halves, Recover Profiles

TRANSFORMATION (PLAIN EXPERIMENT)

Start from the log-ratio view; split, rotate, and recover standard performance profiles.

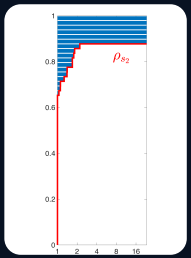


highlighted view



ρ_{s_1} half

+

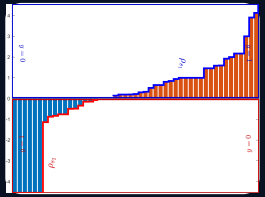


ρ_{s_2} half

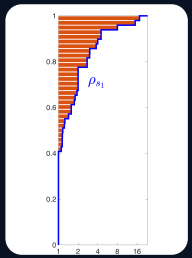
Same Data: Split Halves, Recover Profiles

TRANSFORMATION (PLAIN EXPERIMENT)

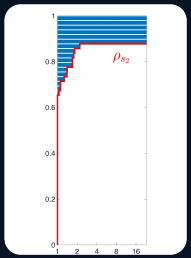
Start from the log-ratio view; split, rotate, and recover standard performance profiles.



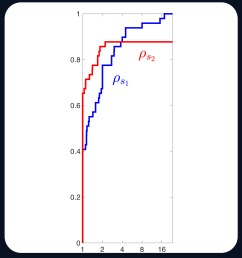
highlighted view



ρ_{s_1} half



ρ_{s_2} half



perf. profiles

Tool: One Function for All

PACKAGE INTERFACE

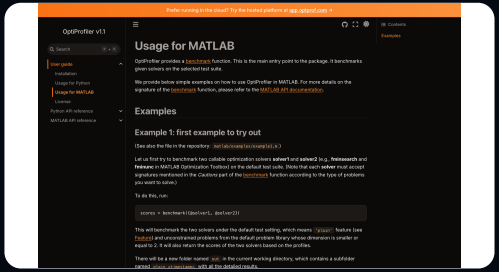
One entry point declares solvers, test suites, feature views, and comparable outputs.

```python
from optiprofiler import benchmark
scores = benchmark(
    [solver1, solver2],
    feature_name='noisy',
)
```

Feature view is a keyword

`feature_name='noisy'` selects the noisy view.

The same call writes profiles, reports, and solver scores.



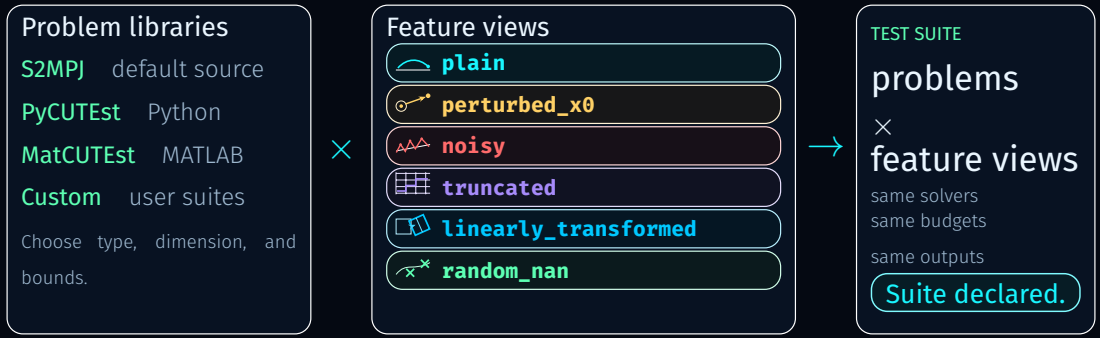
Python API MATLAB API

www.optprof.com

Tool: Building the Test Suite

PROBLEM LIBRARIES × FEATURES

We define a benchmark by crossing real problem sources with controlled feature views.



Tool: Scores Turn Curves Into an Inner Loop

AUTOMATION

Score = average AUC over a solver's own performance profiles in the experiment.

How the score is computed

$$\text{score}(s) = \frac{1}{m_s} \sum_{j=1}^{m_s} \text{AUC}(\rho_{s,j})$$

$\rho_{s,j}$ one generated performance profile for solver s

AUC one scalar from each profile

Average one comparable score per solver

Why quantification matters

Once curves become numbers, they can drive automated search.

Self-tuning

optimize solver settings by score

Code evolution

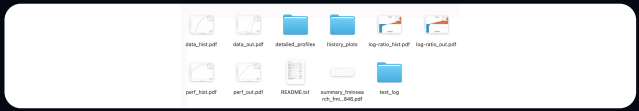
rank generated solver variants

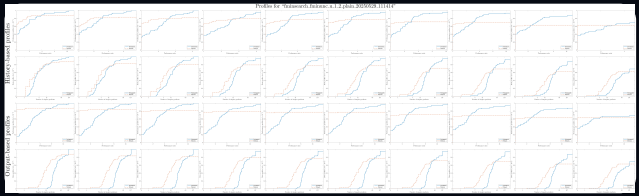
Curves become an objective.

Tool: One Run, All Artifacts

AUTOMATIC OUTPUTS

One benchmark call writes reproducible artifacts and returns solver scores.





Generated automatically

Profiles	perf, data, log-ratio
Summary	all tolerances
Scores	tuning objective
Logs/data	curves, seeds, times

Ready for papers and reruns.

Tool: Profile Details

WHAT GETS PLOTTED

Two cost definitions; repeated runs average performance profiles and show variability.

History-based cost

$$t_{p,s}^{\text{hist}} = \begin{cases} k, & x_{p,s,k} \text{ first recorded point passing the test,} \\ \infty, & \text{none up to } \texttt{maxfun}_p. \end{cases}$$

Output-based cost

$$t_{p,s}^{\text{out}} = \begin{cases} \text{eval}_{p,s}, & x_{p,s}^{\text{out}} \text{ passes the test,} \\ \infty, & \text{otherwise.} \end{cases}$$

Repeated runs: average and shade

With `n_runs`>1, repeat under R seeds.

Average performance profile

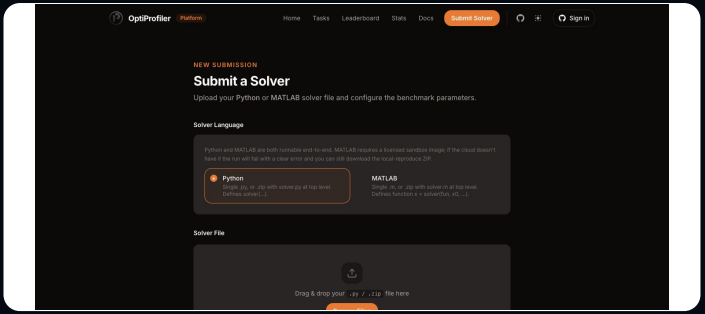
$$\bar{\rho}_s(\alpha) = \frac{1}{R} \sum_{r=1}^{R} \rho_s^{(r)}(\alpha)$$

Min–max error bar

$$\text{band}_s(\alpha) = \left[\min_{1 \le r \le R} \rho_s^{(r)}(\alpha), \max_{1 \le r \le R} \rho_s^{(r)}(\alpha) \right]$$

Platform: Guided Benchmarking Workflow

Guided cloud runs from package upload to shared outputs.



Cloud workflow

app.optprof.com

1. Upload solver
2. Choose suite + budget
3. Submit cloud run

Profiles, logs, scores.

Platform: OPA Across the Journey

BENCHMARK-GROUNDED AGENTS

Let users be lazy: read the docs for them.

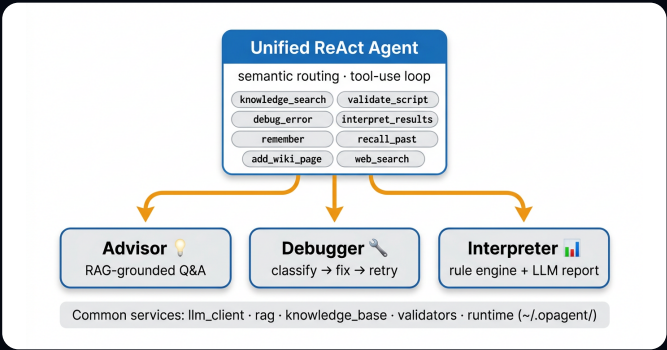
Ship an expert agent.

Q&A usage

Knowledge expert rules

Tools debug runs

Docs become dialogue.



Ecosystem: Open Source Platform + Community Input

COMMUNITY VISION

Shared rules and community input turn tools into an ecosystem.

open-source platform + community input → DFO bench ecosystem

Platform
open code
documented APIs
reproducible outputs

Our input
special structures
hard failures
noisy cases

Shared trust
fair protocols
reporting norms
reusable baselines

Join: AI Lowers Build Cost; Evaluation Is the Bottleneck

PARTICIPATION

AI lowers build cost; evaluation ideas stay scarce.

Problems
libraries
structures
constraints

Features
noise
failures
budgets

Solvers
wrappers
baselines
regression cases

Rules
scoring
credit
reports

Meaningful test suites + reasonable scores make leaderboards possible.

OptiProfiler Ecosystem

Benchmarking Engine • AI Agent • Online Platform for Optimization Solvers

Cunxin Huang • Tom M. Ragonneau • Zaikun Zhang

Automated, reproducible, and insightful benchmarking for DFO solvers

Why it matters

- One-line benchmarking
- Custom testing environments
- Publication-ready visual profiles
- Rigorous & reproducible comparisons

OPA: OptiProfiler Agent

- Advisor: docs & citations
- Debugger: wrappers & failed runs
- Interpreter: explain profiles

```
pip install 'optiprofiler-agent[all]'
```

Features & problem libraries

plain | noisy | perturbed_x0 | hearly_transformed | random_nan | quantized

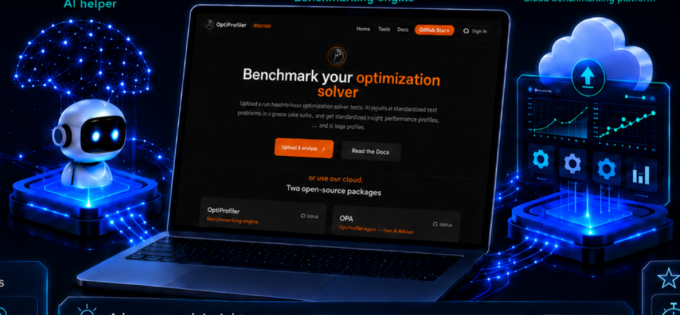
Libraries: S2MPJ, MatCUTEst, PyCUTEst

```
benchmark({@fminsearch, @fminunc})
```

OPA
AI helper

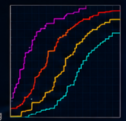
OptiProfiler
Benchmarking engine

app.optprof.com
Cloud benchmarking platform



Core benchmarking tools

- Performance profiles
- Data profiles
- Log-ratio profiles
- History-based vs. output-based costs
- AUC-style solver scoring



Online platform

Upload — secure sandbox → Configure & Run — cloud execution → View Results — results in minutes

A key research insight

Shaded area in a log-ratio profile = area above the performance profile.

A clean bridge between two-solver log ratio profiles and performance profiles.

Technical highlights

- **History- vs. output-based costs** — Choose the right cost model
- **Profiles & AUC scoring** — Performance, data, and log-ratio views
- **Flexible environments** — Plain, noisy, perturbed, quantized
- **Reproducible workflow** — Automated benchmarking from run to report

From local package to cloud platform, OptiProfiler makes optimization benchmarking more accessible, flexible, and reproducible. | Docs: optprof.com | Platform: app.optprof.com

When PDS Fails to Converge

Non-convergence analysis of probabilistic direct search

Non-convergence region

$$m < \log_2\left(1 - \frac{\log\theta}{\log\gamma}\right)$$

$$\sum_{k=0}^{\infty} Y_k \prod_{\ell=0}^{k-1} \gamma^{Y_\ell}\,\theta^{1-Y_\ell}$$

almost tight conditions

Convergence region

$$m > \log_2\left(1 - \frac{\log\theta}{\log\gamma}\right)$$

Core mechanism

random series converges ⇒ iterates remain away from optimum

Friday, June 5, 15:20–15:45

Zaikun Zhang, Sun Yat-sen University

50 George Square G.02

Huang & Zhang · Preprint 2026

Thank you, DFO community!

Any questions?

Acknowledgements

Supervisor Professor Zaikun Zhang

Conference support SIAM Student Travel Award

Doctoral support Hong Kong PhD Fellowship Scheme

Edinburgh, SIAM OP26