

Structured databases for black-box optimization

Collecting and retrieving problems, algorithms, and features

Goals

- Make algorithms, problems, feature sets, ... **findable**
- Connect them with their **properties**
- Provide **foundations** for benchmarking and algorithm selection



openoptimizationorg.github.io/Overview

OPL and Friends: Libraries for optimization problems, algorithms and feature sets

Koen van der Blom, Carola Doerr, Olaf Mersmann, Boris Naujoks, Elena Raponi, Diederick Vermetten, Vanessa Volz and others



OPL

- Optimization **problems, suites and generators**
- **Meta-data** on characteristics and origins

OAL

- Optimization **algorithms + implementations**
- Characteristics and **supported problem features**

OFL

- **Feature sets** which can characterize optimization problems

Use Case: Have a real-world optimization task

Find problems with similar **characteristics**, but faster to evaluate

Find algorithms which can handle the properties of the problem to **start algorithm** development

Find relevant feature sets which can **characterize** the difference between your problem and the ones from OPL

Use Case: Designed a new constraint handling mechanism

Find suitable benchmark problems with **matching** constraint types

Find algorithms to **compare** with, or algorithms to which the method could be added

Find features to characterize the constraint problems to **analyze** the strength and weakness of your method

Use Case: Missing an overview of available resources

Get **statistics** about types of real-world problems available, and what areas are not yet covered by benchmarks

Find what problem properties are not yet **covered** by available optimization algorithms

Find what settings don't yet have easily **accessible** feature sets

OPL How it looks:

Name	Type	Variable Types	Total Variables	Objectives	Properties	Constraint Types	Total Constraints
ATO	Problem	continuous	10	2			
Building spatial design	Problem	binary continuous	>=2	2		unknown box	>=2
Convex DTLZ2	Problem	continuous	>=1	[10, 2, 3, 4, 5, 6, 7, 8, 9]			
Electric Motor Design Optimization	Problem	integer continuous	26	1	noisy	unknown box	>=14
FleetOpt	Problem	integer	{54, 13208}	1		unknown	>=1
Gasoline direct injection engine design	Problem	integer continuous	14	2	multi-fidelity	unknown	5
InverseDeceptiveTrap+RotatedEllipsoid / DeceptiveTrap+RotatedEllipsoid	Problem	binary continuous	>=2	2			
Inverted DTLZ1	Problem	continuous	>=1	[10, 2, 3, 4, 5, 6, 7, 8, 9]			
JSEC2019	Problem	continuous	32	[1, 2, 3, 4, 5]		unknown	22
Onemax+Sphere / DeceptiveTrap+RotatedEllipsoid	Problem	binary continuous	>=2	2			

The screenshot shows a detailed view of a problem in the OPL database. The problem is 'suite_bbob' (BBOB). It is a continuous problem with 10 variables and 1 objective. The interface includes sections for:

- PROPERTIES:** n/a
- CONSTRAINT TYPES:** n/a
- TOTAL CONSTRAINTS:** n/a
- DYNAMICS:** n/a
- NOISE:** n/a
- PARTIAL EVALUATIONS:** n/a
- INDEPENDENT OBJECTIVES:** n/a
- FIDELITY LEVELS:** n/a
- FULL NAME:** n/a
- DESCRIPTION:** n/a
- TAGS:** n/a
- REFERENCES:** COCO: a platform for comparing continuous optimizers in a black-box setting. <https://doi.org/10.1080/10801808927202018089272>
- IMPLEMENTATIONS:** impl_coco
- MODALITY:** multimodal
- EVALUATION TIME:** n/a
- EXAMPLES:** n/a
- SOURCE:** n/a
- BINARY VARS:** n/a
- CATEGORICAL VARS:** n/a
- CONTINUOUS VARS:** >=1
- INTEGER VARS:** n/a
- IMPLEMENTATION NAMES:** COCO framework
- IMPLEMENTATION LANGUAGES:** C/Python
- IMPLEMENTATION EVALUATION TIMES:** n/a
- IMPLEMENTATION LINKS:** <https://github.com/naujoks/coco>
- IMPLEMENTATION DESCRIPTIONS:** Comparing Continuous Optimizers: black-box optimization benchmarking platform
- IMPLEMENTATION REQUIREMENTS:** n/a
- HARD BOX CONSTRAINTS:** n/a
- SOFT BOX CONSTRAINTS:** n/a
- HARD LINEAR CONSTRAINTS:** n/a
- SOFT LINEAR CONSTRAINTS:** n/a
- HARD FUNCTION CONSTRAINTS:** n/a
- SOFT FUNCTION CONSTRAINTS:** n/a

 Below the metadata is a 'README.md' section for 'BBOB', which includes a 'Quick Start' section with instructions to install 'uv' and run the 'call_bbob.py' script. A code snippet is also shown at the bottom of the screenshot.

Community-led effort, we need your help!

- Anything **missing**? Have **corrections**? Share it! Through **GitHub** or Google form
- Properties not covered? **Suggest** them!
- Want to help **build** the tools? Reach out and **join** our meetings! (kvdb@cw.nl)