Benchmarked: Optimization meets Machine Learning

(2020 Lorentz Center Workshop)

Data Formats

Thomas Weise
Institute of Applied Optimization, Hefei University, China
http://iao.hfuu.edu.cn
tweise@ustc.edu.cn • tweise@hfuu.edu.cn

http://iao.hfuu.edu.cn/images/publications/weise_2020_lorentz_data_formats.pdf

Introduction

Experimentation with Optimization Algorithms

- Experimentation is a major driving force in optimization, operations research, AI, machine learning, and so on.

Experimentation with Optimization Algorithms

- Experimentation is a major driving force in optimization, operations research, AI, machine learning, and so on.
- Scientific thoughts determine how we do experiments and how we interpret our results.

Experimentation with Optimization Algorithms

- Experimentation is a major driving force in optimization, operations research, AI, machine learning, and so on.
- Scientific thoughts determine how we do experiments and how we interpret our results.
- I think: The tools we use will influence our decisions very significantly.

Experimentation with Optimization Algorithms

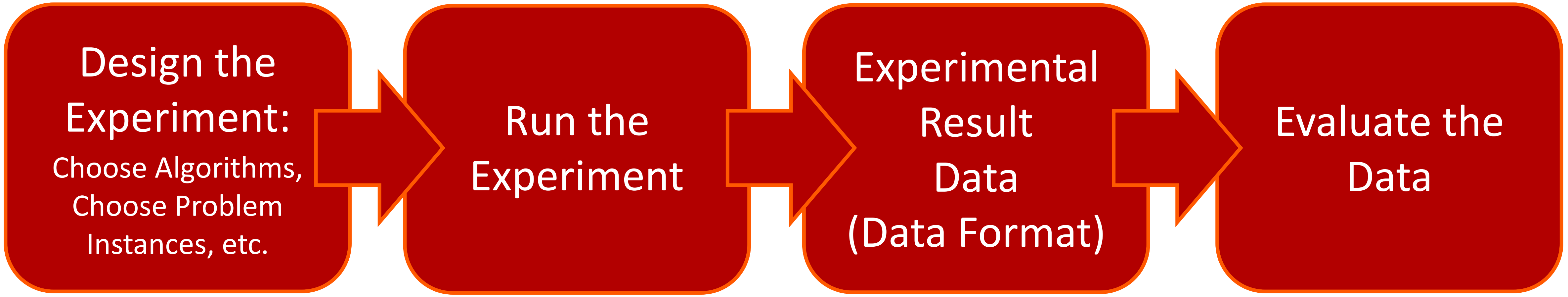
- Experimentation is a major driving force in optimization, operations research, AI, machine learning, and so on.
- Scientific thoughts determine how we do experiments and how we interpret our results.
- I think: The tools we use will influence our decisions very significantly.
- Often, we consider only experimental routes that are technically easy to implement and the data we store results from the routes we take.

Experimentation with Optimization Algorithms

- Experimentation is a major driving force in optimization, operations research, AI, machine learning, and so on.
- Scientific thoughts determine how we do experiments and how we interpret our results.
- I think: The tools we use will influence our decisions very significantly.
- Often, we consider only experimental routes that are technically easy to implement and the data we store results from the routes we take.
- The data format we use for storing the results is an important component of this route.

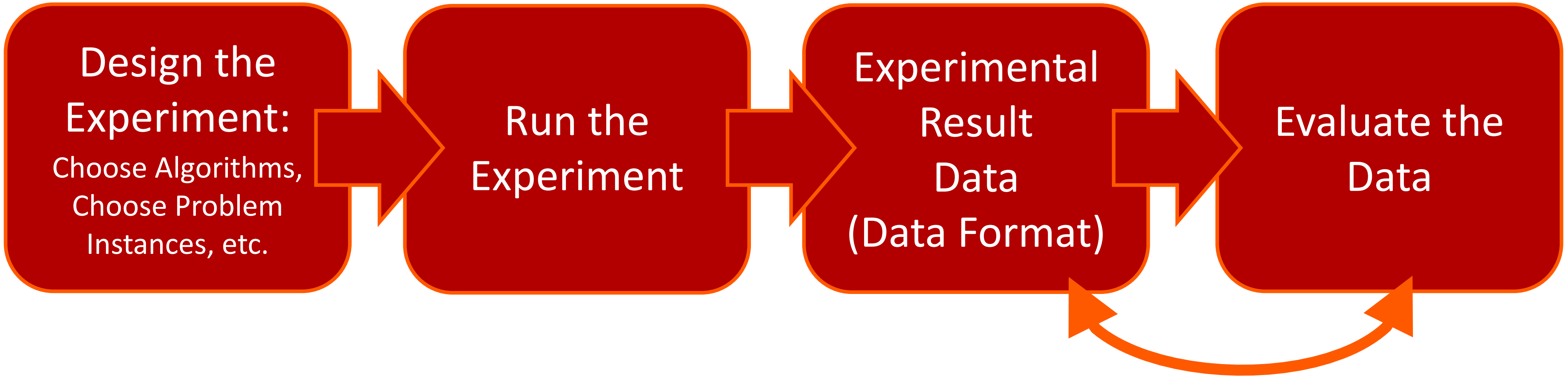
Experimentation with Optimization Algorithms

- Experimental Flow and the Data Format


Design the Experiment:
Choose Algorithms, Choose Problem Instances, etc.
Run the Experiment
Experimental Result Data (Data Format)
Evaluate the Data

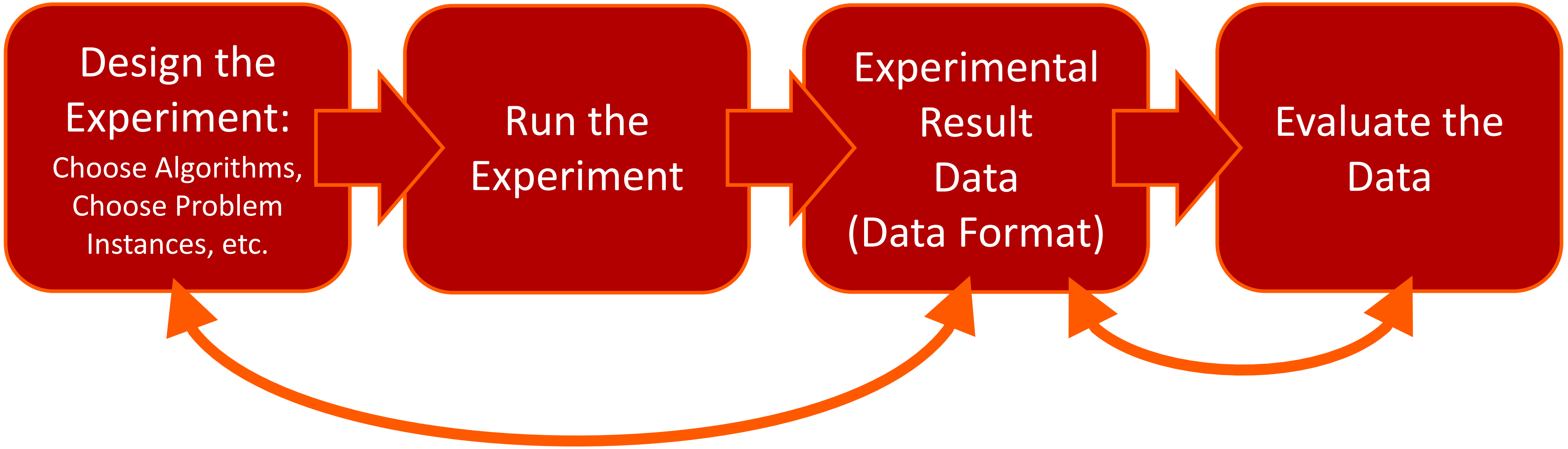
Experimentation with Optimization Algorithms

- Experimental Flow and the Data Format


Design the Experiment:
Choose Algorithms, Choose Problem Instances, etc.
Run the Experiment
Experimental Result Data (Data Format)
Evaluate the Data

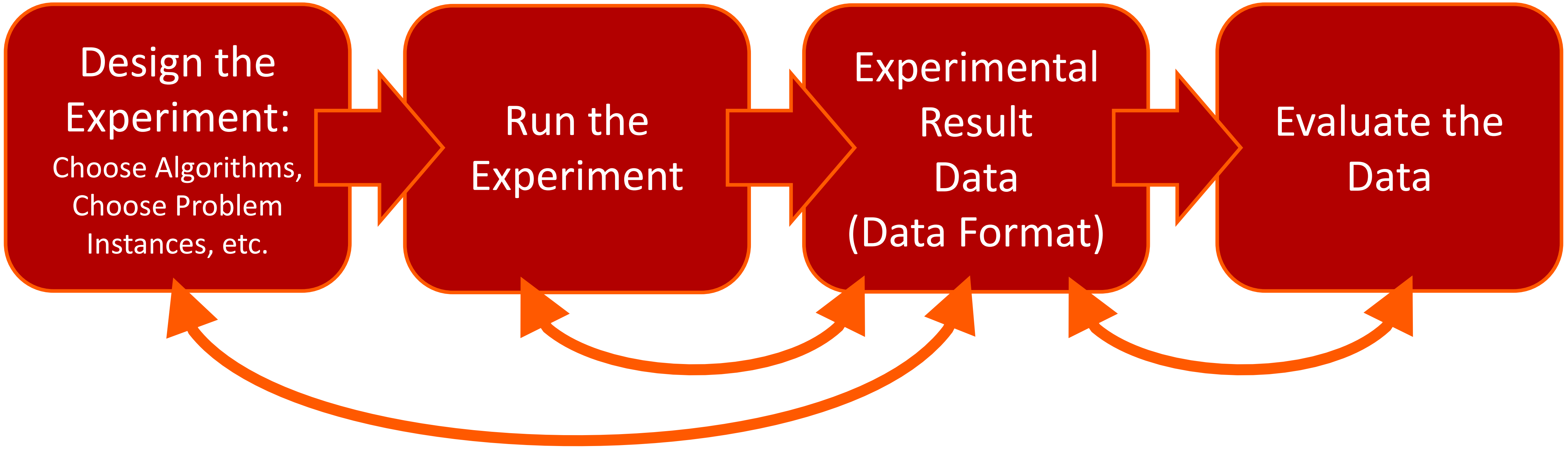
Experimentation with Optimization Algorithms

- Experimental Flow and the Data Format


Design the Experiment:
Choose Algorithms, Choose Problem Instances, etc.
Run the Experiment
Experimental Result Data (Data Format)
Evaluate the Data

Experimentation with Optimization Algorithms

- Experimental Flow and the Data Format


Design the Experiment:
Choose Algorithms, Choose Problem Instances, etc.
Run the Experiment
Experimental Result Data (Data Format)
Evaluate the Data

Goals

- Data should be understandable by others
- Data should be replicable

- Tools for experiments and experiment evaluation (e.g., IOHprofiler, COCO, nevergrad, DSCtool, …) should be inter-operable

Requirements for an Ideal File Format

1. What can we store in one data point?

- We can only analyze and understand metrics that are stored.

1.1. objective function value

- 1.1.1 of current best-so-far candidate solution
- 1.1.2 of last-evaluated candidate solution

1.2. corresponding time stamps

- 1.2.1 clock time, e.g., in milliseconds?
- 1.2.2. in FEs?
- 1.2.3. finer-grained time than FEs (e.g., distance evaluations in TSP, bit flips in SAT)
- 1.2.4. more coarsely grained (such as “generations”)

1.3. the values of adaptive parameters

1.4. other important metrics, such as memory consumption

2. When can we store data points?

- We can only evaluate algorithm behavior at moments for which data points are stored.

2.1. one data point at the end of the run

2.2. one data point for each improvement

2.3. one data point for each FE

2.4. one data point at fixed clock time moments

3. Can results be verified?

- How can we know that the data contained in the file is true?
- Ideally in a machine-readable way, maybe even as code that could be copy-pasted and evaluated with the objective function (if the implementation of our experiment is open source).

3.1. the final solution can be included.

3.2. solutions at special moments can be included

4. Meta-Data and Replicability, Part 1

- Replicability: We can extract all information needed to replicate the experiment?

4.1. Can algorithm name and all parameter values be stored?

4.2. Can problem instance information / objective function be stored such that they can be uniquely identified?

4.3. Can static problem instance information be included? (in case instances cannot be made available, this allows for reproducibility)

4. Meta-Data and Replicability, Part 2

- Can implementation and execution-relevant information be stored?

4.4. random seed

4.5. termination criteria

4.6. things that have static influence on implementation behavior: programming language, compiler, software, and library versions, JVM version, …

4.7. system properties relevant to measurements?

  - Runtime is measured? $\Rightarrow$ store CPU type, speed, motherboard info, RAM size, OS, OS version, JVM version, …
  - GPU is used? $\Rightarrow$ store GPU type

5. Experiment Execution Facilitation

5.1. allows for parallel runs [hard if multiple runs are stored in one file]

5.2. allows for distributed runs

5.3. allows for adding more runs to the same experiment in the future (ideally without the need to repeat the whole experiment from scratch)

5.4. allows for more runs with other instances or other algorithms in the future

5.5. allows for resuming experiments that have been aborted or where processes failed (without the need to repeat the whole experiment from scratch)

6. Choice of Framework Format

6.1. databases could be used

- personal experience: this does not scale well, not a good long-term choice, backup - export to text formats such as SQL anyway

6.2. binary data

- smaller file size, but low long-term utilization (need lots of documentation)

6.3. human-readable text formats.

- self-documenting and allows unrelated researchers to interpret results who, in a few years from now do not have our software or program source
- ideally formats such as plain `text`, `csv`, XML, JSON
- can easily be compressed, e.g., with `tar.xz` down to **0.3%** of the text size…

7. Human-Understandable Structure

- What is the overall layout of the data?

7.1. Is all data concerning one run in one single file?

7.2. Are meta-data and experimental results distributed over multiple files?

(the first leads to bigger files – maybe no problem when compressed – while the second creates the danger of partial data loss.)

7.3. If complex experiments are run for multiple instances and algorithm setups are run, then is the data structured according

7.3.1. to fixed rules and

7.3.2. in a way that is easy to understand for someone not knowing the rules

8. Software Support

8.1. Do we have open source software that read and write the file format?

8.2. Do we have open source software that can convert the file format to another format?

8.3. Do we have open source software that can extract basic statistics from the file format?

(Bonus points for availability in different programming languages.)

Existing File Formats

A. Plain Comma-Separated-Values

- With column headlines, allows for storing objective values, arbitrary time measurements, and for the values of adaptive parameters at arbitrary moments during the runs
- If candidate solutions have linear representations, they can be stored in additional columns
- Human readable and broad software support
- Cannot store meta-data efficiently (putting meta-data into an additional text file per setup may be a good solution)
- No pre-defined structure for complex experiments
- 1 run = 1 file allows arbitrary distribution and parallelization of experiments as well as later adding more experiments and continuation of aborted experiments

B. The COCO Format

- human-readable format with understandable folder structure, partially CSV-based
- can log either end quality, quality at every FE, quality on improvement
- can provide full final/intermediate solutions
- measures time in FEs
- stores benchmark function dimension and ID (docu needed to understand)
- termination criteria (e.g., goal quality) can be stored in index file
- algorithm setup can be stored as unstructured comment in index file
- resuming experiments is possible, but requires manually editing of files

N. Hansen, A. Auger, S. Finck, and R. Ros. *Real-Parameter Black-Box Optimization Benchmarking: Experimental Setup*. November 2015.
http://coco.lri.fr/downloads/download15.03/bbobdocexperiment.pdf

C. AITOA Format

- 1 file = 1 run, 1 file contains complete information about setup
- (time, FEs, result) tuples in CSV format
- keys-value pairs regarding system setup, termination criteria, random seed, algorithm configuration, problem information, software version, etc.
- end results contained (in search and solution space)
- hierarchical folder structure algorithmName -> instanceName -> run

Thank you!

谢谢您们!

http://iao.hfuu.edu.cn/images/publications/weise_2020_lorentz_data_formats.pdf