中国科学技术大学
University of Science and Technology of China

The optimizationBenchmarking.org Experiment Evaluator

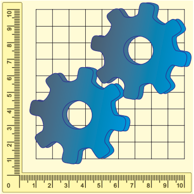
Thomas Weise
tweise@ustc.edu.cn · tweise@gmx.de · http://www.it-weise.de

USTC-Birmingham Joint Res. Inst. in Intelligent Computation and Its Applications (UBRI)
University of Science and Technology of China (USTC), Hefei 230027, Anhui, China

September 14, 2015

Outline

1. Introduction
2. Example 1: MAX-SAT
3. Example 2: BBOB
4. Conclusions



Visit our website

`http://www.optimizationBenchmarking.org`

or

`http://optimizationbenchmarking.github.io/optimizationBenchmarking`

for downloading the software (version 0.8.4) and obtaining more information.

System Requirements:

- Java 1.7 (Ideally a `JDK`, under `JRE` slower with more memory requirements)
- optional: a LaTeX installation, such as TeXLive or MiKTeX (needed for generating pdf reports)

Highlights

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data
3. Comprehensive result and comparison reports with various diagrams and performance metrics, (almost) ready-to-use for publications

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data
3. Comprehensive result and comparison reports with various diagrams and performance metrics, (almost) ready-to-use for publications
4. Diagrams and evaluation criteria can freely be chosen (amongst implemented modules)

Highlights

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data
3. Comprehensive result and comparison reports with various diagrams and performance metrics, (almost) ready-to-use for publications
4. Diagrams and evaluation criteria can freely be chosen (amongst implemented modules)
5. Results can be grouped according to benchmark instance features and/or algorithm parameters

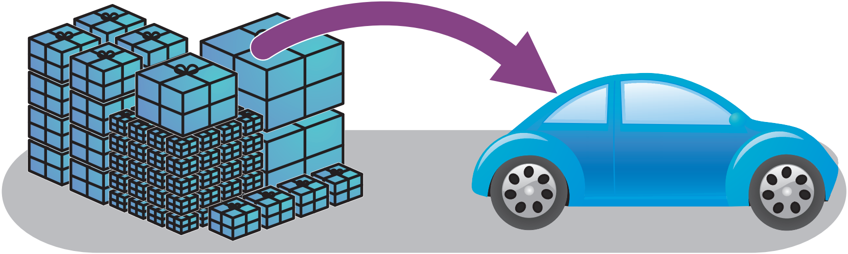
1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data
3. Comprehensive result and comparison reports with various diagrams and performance metrics, (almost) ready-to-use for publications
4. Diagrams and evaluation criteria can freely be chosen (amongst implemented modules)
5. Results can be grouped according to benchmark instance features and/or algorithm parameters
6. Produces either XHTML web pages, LaTeX documents (for several different standard conference or article document classes), or exports results

1. `optimizationBenchmarking` tool for evaluating and comparing experimental results of optimization or Machine Learning algorithms
2. Can easily be configured to load virtually arbitrary experimental result data
3. Comprehensive result and comparison reports with various diagrams and performance metrics, (almost) ready-to-use for publications
4. Diagrams and evaluation criteria can freely be chosen (amongst implemented modules)
5. Results can be grouped according to benchmark instance features and/or algorithm parameters
6. Produces either XHTML web pages, LaTeX documents (for several different standard conference or article document classes), or exports results
7. Easily extensible: Add your own evaluation modules for your own, maybe problem-specific statistics

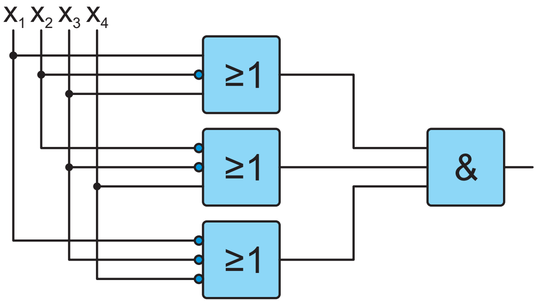
- Many questions in the real world are actually optimization problems

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit certain set of cities in China and return to Hefei!

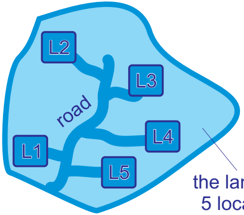
- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit certain set of cities
  - I need to transport n items from here to Feixi but they are too big to transport them all at once. How can I load them best into my car so that I have to travel back and forth the least times?



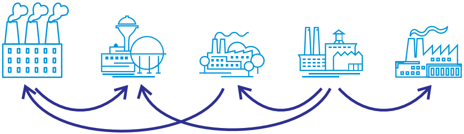
- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit certain set of cities
  - I need to transport n items from here to Feixi
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ become true (or, at least, as *many* of its terms as possible)?



- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit certain set of cities
  - I need to transport n items from here to Feixi
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ become true
  - I want to build a large factory with n workshops. I know the flow of material between each two workshops and now need to choose the locations of the workshops such that the overall running cost incurred by material transportation is *minimized*.

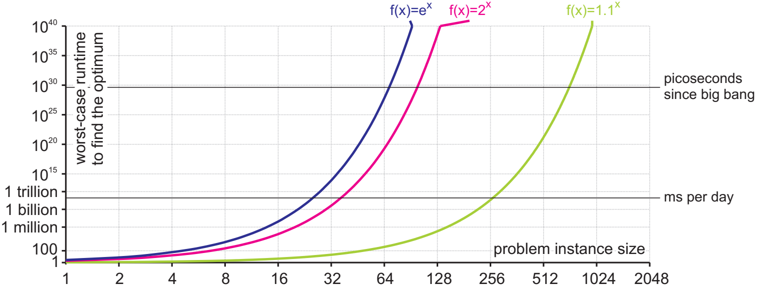


the land with
5 locations

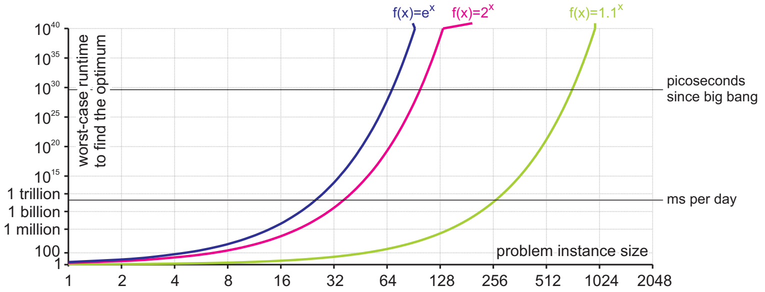


5 workshops and goods flows betwee them
which need to be assigned to locations

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit certain set of cities
  - I need to transport n items from here to Feixi
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \lor \neg x_2 \lor x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)$ become true
  - I want to build a large factory with n workshops.
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.



- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.



- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19]

Optimization Algorithms

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47] or Tabu Search [48–52]

- Many questions in the real world are actually optimization problems, e.g.,
    - Traveling Salesman Problem [60–63]
    - Bin Packing Problem [64]
    - Maximum (3-)Satisfiability Problem [65–68]
    - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47] or Tabu Search [48–52], as well as hybrids of local and global search, such as Memetic Algorithms [53–59]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are... many
- Which of them is best (for my problem)?

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are... many
- Which of them is best (for my problem)?
- How can I make a good algorithm better (for my problem)?

- Which of the algorithms is best (for my problem)?

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}\left(n^2\right)$ *steps to sort* n *elements in the average case."*

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps to sort n elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}\left(n^2\right)$ *steps to sort* n *elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps to sort* n *elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
    - analysis extremely complicated

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps to sort* n *elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps to sort n elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size)

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps to sort n elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps to sort n elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}\left(n^2\right)$ *steps to sort* n *elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - theoretical results only available for toy problems and extremely simplified algorithms.

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}\left(n^2\right)$ steps to sort n elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - theoretical results only available for toy problems and extremely simplified algorithms.
  - Currently, not mature enough to be an easy-to-use tool for practitioners

- Which of the algorithms is best (for my problem)?
- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps to sort* n *elements in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - theoretical results only available for toy problems and extremely simplified algorithms.
  - Currently, not mature enough to be an easy-to-use tool for practitioners
- Experimental analysis and comparison only practical alternative.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]:

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

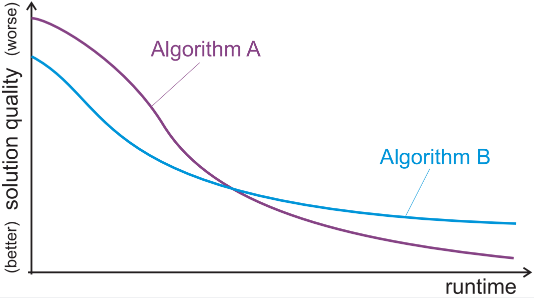
- Algorithm performance has two dimensions [71, 72]: solution quality

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime
- Anytime Algorithms [73] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.



"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime
- Anytime Algorithms [73] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime
- Anytime Algorithms [73] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime
- Anytime Algorithms [73] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime
- Anytime Algorithms [73] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.
- Experiments must capture solution quality and runtime data.

- In optimization or Machine Learning, the following experimental procedure is often used

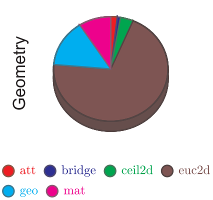
- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a benchmark instance

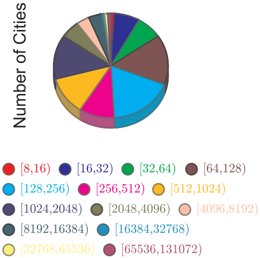
- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances:
     - multiple instances

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances:
     - multiple instances
     - which cover some different problem features

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances:
     - multiple instances
     - which cover some different problem features
     - should be well-known to make results comparable

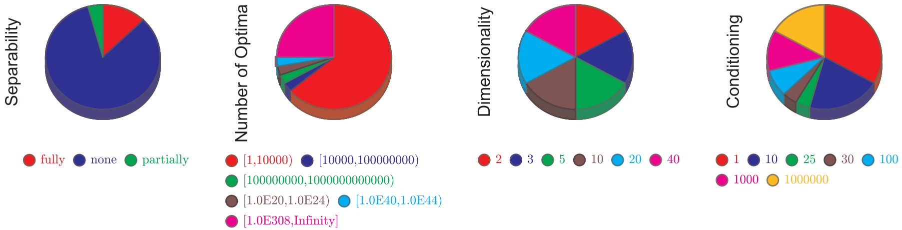
- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances:
     - multiple instances
     - which cover some different problem features
     - should be well-known to make results comparable
     - e.g., *TSPLib*[77–79] for the TSP has instances with different numbers of cities and geometries





The relative amounts of the instances of the 110 symmetric instances of *TSPLib* according to their features (the 10 asymmetric instances are not plotted).

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances:
     - multiple instances
     - which cover some different problem features
     - should be well-known to make results comparable
     - e.g., *BBOB*[71, 80–82] offers different benchmark functions for numerical optimization problems



The relative amounts of *BBOB* benchmark functions according to their features.

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiment

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiment:
     - conduct several independent runs of algorithm for each benchmark instance

Experimental Procedure

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiment:
     - conduct several independent runs of algorithm for each benchmark instance
     - collect algorithm progress informatio, e.g., as *"runtime bestObjectiveValue"* tuples

1	39334	3	42.19354838709677	4075
2	311078	5	70.32258064516128	3976
3	311078	5	70.32258064516128	3894
4	311078	5	70.32258064516128	3824
5	311078	5	70.32258064516128	3761
6	311078	5	70.32258064516128	3705
...				
24099	1111598495	11393	160237.03225806452	2579

FEs: function evaluations
DEs: accesses to distance matrix
AT: absolute runtime in ms
NT: absolute runtime divided by machine-specific performance factor
best objective value: best result so far

Example for data collected in a log file by *TSP Suite*[72, 83].

- In optimization or Machine Learning, the following experimental procedure is often used
    1. Select a set of benchmark instances
    2. Do experiment:
        - conduct several independent runs of algorithm for each benchmark instance
        - collect algorithm progress informatio, e.g., as *"runtime bestObjectiveValue"* tuples
        - one log file per run, each log file has several such tuples

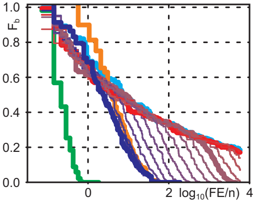
- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments:
     - conduct several independent runs of algorithm for each benchmark instance
     - collect algorithm progress informatio, e.g., as *"runtime bestObjectiveValue"* tuples
     - one log file per run, each log file has several such tuples
     - repeat for different algorithm parameter settings (e.g., different population sizes of an EA)

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments:
     - conduct several independent runs of algorithm for each benchmark instance
     - collect algorithm progress informatio, e.g., as *"runtime bestObjectiveValue"* tuples
     - one log file per run, each log file has several such tuples
     - repeat for different algorithm parameter settings (e.g., different population sizes of an EA)
     - repeat with other algorithms for comparison purposes

Experimental Procedure

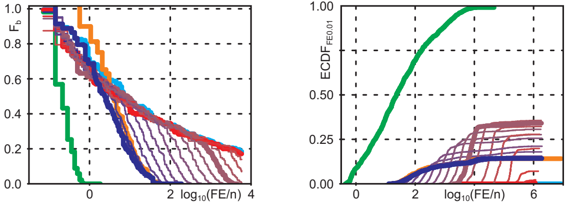
- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time



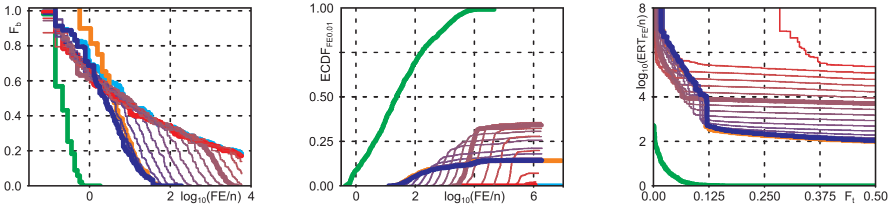
Examples for progress diagrams for different algorithms (signified by different colors) over different sub-sets of the *TSPLib* data.

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] (over time)



Examples for progress and ERT diagrams for different algorithms (signified by different colors) over different sub-sets of the *TSPLib* data.

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)



Examples for progress, ERT, and ECDF diagrams for different algorithms (signified by different colors) over different sub-sets of the *TSPLib* data.

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)
     - use statistical tests to compare results (at different points during the runs)

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)
     - use statistical tests to compare results (at different points during the runs)
     - analyze the impact of benchmark features and algorithm parameters on the above

Experimental Procedure

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)
     - use statistical tests to compare results (at different points during the runs)
     - analyze the impact of benchmark features and algorithm parameters on the above
  4. Draw conclusions about algorithm performance and parameter settings

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)
     - use statistical tests to compare results (at different points during the runs)
     - analyze the impact of benchmark features and algorithm parameters on the above
  4. Draw conclusions about algorithm performance and parameter settings
  5. But this is all *very* cumbersome, involves much work and much data...

- In optimization or Machine Learning, the following experimental procedure is often used
  1. Select a set of benchmark instances
  2. Do experiments
  3. Evaluate the gathered data:
     - draw diagrams of progress of solution quality over time
     - draw diagrams of advanced statistical parameters such as ECDF [66, 72, 80, 84] and ERT [72, 80] (over time)
     - use statistical tests to compare results (at different points during the runs)
     - analyze the impact of benchmark features and algorithm parameters on the above
  4. Draw conclusions about algorithm performance and parameter settings
  5. But this is all *very* cumbersome, involves much work and much data...
- The `optimizationBenchmarking` Evaluator can automatize much of this work

1. Introduction
2. Example 1: MAX-SAT
3. Example 2: BBOB
4. Conclusions

- So much about theory.

- So much about theory.
- But what is this "`optimizationBenchmarking`" and what can it do for me?

- So much about theory.
- But what is this "`optimizationBenchmarking`" and what can it do for me?
- Let us look at how research and experimentation on optimization or Machine Learning can work on a practical example.

- So much about theory.
- But what is this "`optimizationBenchmarking`" and what can it do for me?
- Let us look at how research and experimentation on optimization or Machine Learning can work on a practical example.
- Assume that we are a researcher working on the MAX-3SAT problem, with new and fresh ideas...

- Satisfiability Problems

- Satisfiability Problems
  - The satisfiability problem (SAT) is one of the most prominent problems in artificial intelligence, logic, theoretical computer science, and various application areas. [65]

- Satisfiability Problems
  - The satisfiability problem (SAT) is one of the most prominent problems in artificial intelligence, logic, theoretical computer science, and various application areas. [65]
  - Given: formula B in Boolean logic consisting of n Boolean variables $\vec{x} = (x_1, x_2, \dots, x_n)^T$ which each can be either `true` or `false`

- Satisfiability Problems
  - The satisfiability problem (SAT) is one of the most prominent problems in artificial intelligence, logic, theoretical computer science, and various application areas. [65]
  - Given: formula B in Boolean logic consisting of n Boolean variables $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ which each can be either `true` or `false`
  - Goal: find a setting for these variables so that B becomes `true`

- Satisfiability Problems
- CNF 3-SAT Problems

- Satisfiability Problems
- CNF 3-SAT Problems
  - B consists of k clauses $C_1 \dots C_k$

$$\underbrace{B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}) \wedge \dots}_{k \text{ clauses } (C_1 \dots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
  - B consists of k clauses $C_1 \dots C_k$
  - each clause consists of 3 literals

$$B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{\underbrace{(x_7 \lor x_4 \lor \neg x_2)}_{1 \text{ clause } (C_1)} \land (\underbrace{\neg x_7}_{} \lor \neg x_4 \lor x_3) \land (\underbrace{x_x \lor \neg x_1 \lor x_2}_{3 \text{ literals in } 1 \text{ clause}}) \land \dots}_{k \text{ clauses } (C_1 \dots C_k)} \quad (1)$$

- Satisfiability Problems
- CNF 3-SAT Problems
  - B consists of k clauses $C_1 \dots C_k$
  - each clause consists of 3 literals
  - a literal can either be a variable (e.g., x_5) or its negate (e.g., $\neg x_5$)

$$B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7}_{1 \text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3 \text{ literals in } 1 \text{ clause}}) \wedge \dots}_{k \text{ clauses } (C_1 \dots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
  - B consists of k clauses $C_1 \dots C_k$
  - each clause consists of 3 literals
  - a literal can either be a variable (e.g., x_5) or its negate (e.g., $\neg x_5$)
  - in a clause, the 3 literals are combined with logical *or* ($\vee$)

$$B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7}_{1 \text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3 \text{ literals in } 1 \text{ clause}}) \wedge \dots}_{k \text{ clauses } (C_1 \dots C_k)} \qquad (1)$$

- Satisfiability Problems
- CNF 3-SAT Problems
  - B consists of k clauses $C_1 \dots C_k$
  - each clause consists of 3 literals
  - a literal can either be a variable (e.g., x_5) or its negate (e.g., $\neg x_5$)
  - in a clause, the 3 literals are combined with logical *or* ($\vee$)
  - in the formula B, all k clauses are combined with logical *and* ($\wedge$)

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1\text{ clause }(C_1)} \wedge (\underbrace{\neg x_7}_{1\text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3\text{ literals in }1\text{ clause}}) \wedge \dots}_{k\text{ clauses }(C_1 \dots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \lor x_4 \lor \neg x_2)}_{1\text{ clause }(C_1)} \land (\underbrace{\neg x_7}_{1\text{ literal}} \lor \neg x_4 \lor x_3) \land (\underbrace{x_x \lor \neg x_1 \lor x_2}_{3\text{ literals in }1\text{ clause}}) \land \dots}_{k\text{ clauses }(C_1 \dots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1\text{ clause }(C_1)} \wedge (\underbrace{\neg x_7}_{1\text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3\text{ literals in }1\text{ clause}}) \wedge \dots}_{k\text{ clauses }(C_1 \dots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1\text{ clause }(C_1)} \wedge (\underbrace{\neg x_7}_{1\text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3\text{ literals in }1\text{ clause}}) \wedge \dots}_{k\text{ clauses }(C_1 \dots C_k)} \quad (1)$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible
  - if all are `true` $\Longrightarrow B$ is satisfied

$$\underbrace{B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7}_{1 \text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3 \text{ literals in } 1 \text{ clause}}) \wedge \dots}_{k \text{ clauses } (C_1 \dots C_k)} \quad (1)$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible
  - if all are `true` $\Longrightarrow B$ is satisfied
  - define objective function $f(\vec{x}) = \#$ clauses which are `false`

$$B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7}_{1 \text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3 \text{ literals in } 1 \text{ clause}}) \wedge \ldots}_{k \text{ clauses } (C_1 \ldots C_k)} \tag{1}$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible
  - if all are `true` $\Longrightarrow B$ is satisfied
  - define objective function $f(\vec{x}) = \#$ clauses which are `false`
  - $f(\vec{x}) = 0 \Longrightarrow$ all clauses are `true`

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1\text{ clause }(C_1)} \wedge (\underbrace{\neg x_7}_{1\text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3\text{ literals in }1\text{ clause}}) \wedge \ldots}_{k\text{ clauses }(C_1 \ldots C_k)} \quad (1)$$

中国科学技术大学 University of Science and Technology of China

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible
  - if all are `true` $\Longrightarrow$ B is satisfied
  - define objective function $f(\vec{x}) = \#$ clauses which are `false`
  - $f(\vec{x}) = 0 \Longrightarrow$ all clauses are `true`
  - $f(\vec{x}) = k \Longrightarrow$ all clauses are `false`

$$B(\underbrace{\vec{x}}_{n \text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1 \text{ clause } (C_1)} \wedge (\underbrace{\neg x_7}_{1 \text{ literal}} \vee \neg x_4 \vee x_3) \wedge \underbrace{(x_x \vee \neg x_1 \vee x_2)}_{3 \text{ literals in } 1 \text{ clause}} \wedge \dots}_{k \text{ clauses } (C_1 \dots C_k)} \quad (1)$$

- Satisfiability Problems
- CNF 3-SAT Problems
- MAX-3SAT
  - CNF 3-SAT turned into an optimization problem [36]
  - make as many clauses become `true` as possible
  - if all are `true` $\Longrightarrow B$ is satisfied
  - define objective function $f(\vec{x}) = \#$ clauses which are `false`
  - $f(\vec{x}) = 0 \Longrightarrow$ all clauses are `true`
  - $f(\vec{x}) = k \Longrightarrow$ all clauses are `false`
  - $k + 1$ different objective values possible

$$B(\underbrace{\vec{x}}_{n\text{ variables}}) = \underbrace{\underbrace{(x_7 \vee x_4 \vee \neg x_2)}_{1\text{ clause }(C_1)} \wedge (\underbrace{\neg x_7}_{1\text{ literal}} \vee \neg x_4 \vee x_3) \wedge (\underbrace{x_x \vee \neg x_1 \vee x_2}_{3\text{ literals in }1\text{ clause}}) \wedge \dots}_{k\text{ clauses }(C_1 \dots C_k)} \quad (1)$$

- We want to compare the performance of six algorithms

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
     - starts with random bit string

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
     - starts with random bit string
     - in each iteration flips a randomly chosen bit

Investigated Algorithms

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
     - starts with random bit string
     - in each iteration flips a randomly chosen bit
     - if new solution is better, keep it

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
     - starts with random bit string
     - in each iteration flips a randomly chosen bit
     - if new solution is better, keep it
     - otherwise, undo change

Investigated Algorithms

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
     - same as 1-flip Hill Climber, but

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
     - same as 1-flip Hill Climber, but
     - restart if no improvement after z steps

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
     - same as 1-flip Hill Climber, but
     - restart if no improvement after z steps
     - $z = 1$ at beginning, increased by 1 at each restart

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
      - like 1-flip Hill Climber, but

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
     - like 1-flip Hill Climber, but
     - in each iteration flips one *or two* randomly chosen bits

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
     - like 1- or 2-flip Hill Climber, but

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
     - like 1- or 2-flip Hill Climber, but
     - in each iteration, randomly chose m bits to flip (m chosen according to a geometric distribution)

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
     - like 1- or 2-flip Hill Climber, but
     - in each iteration, randomly chose m bits to flip (m chosen according to a geometric distribution)
     - if new solution is better, keep it, otherwise undo change

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
     - like 1- or 2-flip Hill Climber, but
     - in each iteration, randomly chose m bits to flip (m chosen according to a geometric distribution)
     - if new solution is better, keep it, otherwise undo change
     - all other bits must have been chosen once before a given bit can be chosen again

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
  6. m-flip Hill Climber with Restarts

- We want to compare the performance of six algorithms:
  1. 1-flip Hill Climber
  2. 1-flip Hill Climber with Restarts
  3. 2-flip Hill Climber
  4. 2-flip Hill Climber with Restarts
  5. m-flip Hill Climber
  6. m-flip Hill Climber with Restarts
- Which of these algorithms performs best? When? Why?

- As benchmark, we use *some* instances from *SATLib* [65]

- As benchmark, we use *some* instances from *SATLib*[65]:

Instance Set	n	k	Instance Set	n	k
uf020	20	91	uf150	150	645
uf050	50	218	uf175	175	753
uf075	75	325	uf200	200	860
uf100	100	430	uf225	225	960
uf125	125	538	uf250	250	1065

- As benchmark, we use *some* instances from *SATLib*[65]:

Instance Set	n	k	Instance Set	n	k
uf020	20	91	uf150	150	645
uf050	50	218	uf175	175	753
uf075	75	325	uf200	200	860
uf100	100	430	uf225	225	960
uf125	125	538	uf250	250	1065

- We pick the first ten instances from each set, i.e., test 100 instances in total

- As benchmark, we use *some* instances from *SATLib* [65]:

Instance Set	n	k	Instance Set	n	k
uf020	20	91	uf150	150	645
uf050	50	218	uf175	175	753
uf075	75	325	uf200	200	860
uf100	100	430	uf225	225	960
uf125	125	538	uf250	250	1065

- We pick the first ten instances from each set, i.e., test 100 instances in total
- All instances are satisfiable

- As benchmark, we use *some* instances from *SATLib* [65]:

Instance Set	n	k	Instance Set	n	k
uf020	20	91	uf150	150	645
uf050	50	218	uf175	175	753
uf075	75	325	uf200	200	860
uf100	100	430	uf225	225	960
uf125	125	538	uf250	250	1065

- We pick the first ten instances from each set, i.e., test 100 instances in total
- All instances are satisfiable
- The problem instances have the following features

- As benchmark, we use *some* instances from *SATLib* [65]:

Instance Set	n	k	Instance Set	n	k
`uf020`	20	91	`uf150`	150	645
`uf050`	50	218	`uf175`	175	753
`uf075`	75	325	`uf200`	200	860
`uf100`	100	430	`uf225`	225	960
`uf125`	125	538	`uf250`	250	1065

- We pick the first ten instances from each set, i.e., test 100 instances in total
- All instances are satisfiable
- The problem instances have the following features:
  - n: the number of variables

- As benchmark, we use *some* instances from *SATLib* [65]:

Instance Set	$\mathbf{n}$	$\mathbf{k}$	Instance Set	$\mathbf{n}$	$\mathbf{k}$
`uf020`	20	91	`uf150`	150	645
`uf050`	50	218	`uf175`	175	753
`uf075`	75	325	`uf200`	200	860
`uf100`	100	430	`uf225`	225	960
`uf125`	125	538	`uf250`	250	1065

- We pick the first ten instances from each set, i.e., test 100 instances in total
- All instances are satisfiable
- The problem instances have the following features:
  - n: the number of variables
  - k: the number of clauses (related to n)

- Now we want to do the experiments.

- Now we want to do the experiments.
- What data shall we collect?

Experiments

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)
  3. $k+1$ possible objective values $\Longrightarrow$ at most $k+2$ log points

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)
  3. $k+1$ possible objective values $\Longrightarrow$ at most $k+2$ log points
  4. In each log point we record

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)
  3. $k + 1$ possible objective values $\Longrightarrow$ at most $k + 2$ log points
  4. In each log point we record
     - the number of function evaluations (*FEs*) performed

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)
  3. $k+1$ possible objective values $\Longrightarrow$ at most $k+2$ log points
  4. In each log point we record
     - the number of function evaluations (*FEs*) performed
     - the ellapsed runtime *RT* (in ns)

- Now we want to do the experiments.
- What data shall we collect?
  1. Data should allow us to reproduce algorithm progress over time
  2. We can collect one data point whenever the algorithm makes an improvement in terms of f (and one at the end of run)
  3. $k+1$ possible objective values $\Longrightarrow$ at most $k+2$ log points
  4. In each log point we record
     - the number of function evaluations (*FEs*) performed
     - the ellapsed runtime *RT* (in ns)
     - the best objective value F achieved so far

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2nd benchmark instance of set uf075.

Listing: Log File uf075-02_2FlipHCrs_01.txt.

```
1        9806        46
3        24643       28
17       106040      25
19       115529      23
20       120373      21
25       144087      18
31       172967      16
290      1550118     15
296      1576034     14
297      1579525     13
300      1592492     12
323      1692189     10
332      1732127     9
1082     5436999     8
1558     7670059     7
2008     9765759     6
2024     9830168     5
2809     13302012    4
5246     24105640    3
6330     28508740    2
17284    73166926    1
60865    238968738   0
```

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2nd benchmark instance of set `uf075`.

Listing: Log File `uf075-02_2FlipHCrs_01.txt`.

log point

1	9806	46
3	24643	28
17	106040	25
19	115529	23
20	120373	21
25	144087	18
31	172967	16
290	1550118	15
296	1576034	14
297	1579525	13
300	1592492	12
323	1692189	10
332	1732127	9
1082	5436999	8
1558	7670059	7
2008	9765759	6
2024	9830168	5
2809	13302012	4
5246	24105640	3
6330	28508740	2
17284	73166926	1
60865	238968738	0

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2nd benchmark instance of set uf075.

Listing: Log File uf075-02_2FlipHCrs_01.txt.

log point

ellapsed *FEs*

1	9806	46
3	24643	28
17	106040	25
19	115529	23
20	120373	21
25	144087	18
31	172967	16
290	1550118	15
296	1576034	14
297	1579525	13
300	1592492	12
323	1692189	10
332	1732127	9
1082	5436999	8
1558	7670059	7
2008	9765759	6
2024	9830168	5
2809	13302012	4
5246	24105640	3
6330	28508740	2
17284	73166926	1
60865	238968738	0

Example of Log File

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2nd benchmark instance of set `uf075`.

Listing: Log File `uf075-02_2FlipHCrs_01.txt`.

log point

ellapsed *FEs*

runtime [ns]

```
1        9806        46
3        24643       28
17       106040      25
19       115529      23
20       120373      21
25       144087      18
31       172967      16
290      1550118     15
296      1576034     14
297      1579525     13
300      1592492     12
323      1692189     10
332      1732127     9
1082     5436999     8
1558     7670059     7
2008     9765759     6
2024     9830168     5
2809     13302012    4
5246     24105640    3
6330     28508740    2
17284    73166926    1
60865    238968738   0
```

Example of Log File

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2nd benchmark instance of set `uf075`.

Listing: Log File `uf075-02_2FlipHCrs_01.txt`.

```
1       9806        46
3       24643       28
17      106040      25
19      115529      23
20      120373      21
25      144087      18
31      172967      16
290     1550118     15
296     1576034     14
297     1579525     13
300     1592492     12
323     1692189     10
332     1732127     9
1082    5436999     8
1558    7670059     7
2008    9765759     6
2024    9830168     5
2809    13302012    4
5246    24105640    3
6330    28508740    2
17284   73166926    1
60865   238968738   0
```

log point

ellapsed *FEs*

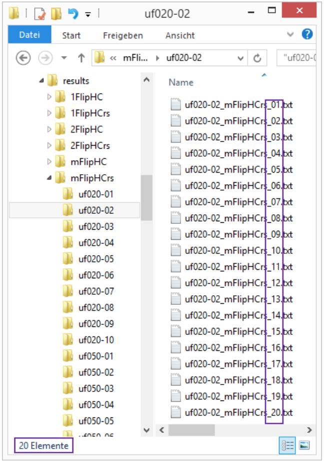
runtime [ns]

F: best $f(\vec{x})$

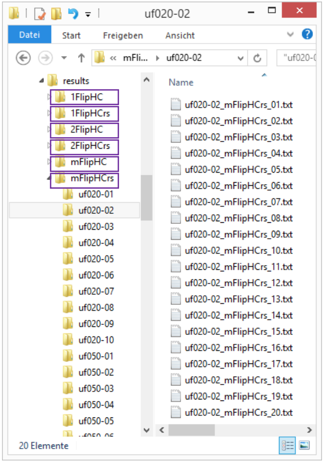
- OK, so after the experiment...

Obtained Data

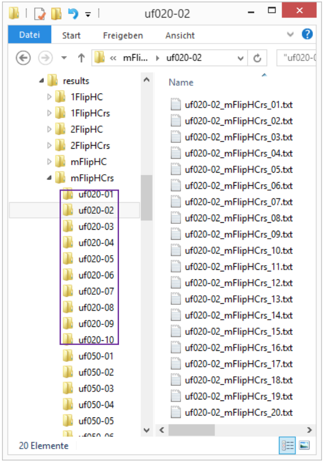
- OK, so after the experiment. . .
  - . . . we have 20 independent runs (log files)



- OK, so after the experiment. . .
  - . . . we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,

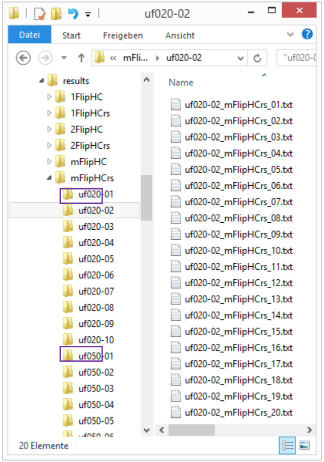


- OK, so after the experiment...
  - ...we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,
  - on each of the 10 benchmark instances

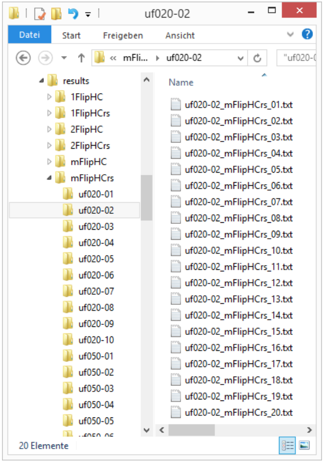


Obtained Data

- OK, so after the experiment. . .
  - . . . we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,
  - on each of the 10 benchmark instances
  - of each of the 10 instance sets.

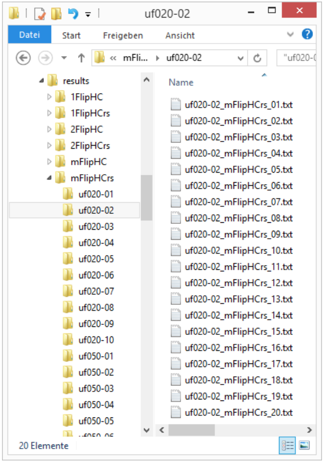


- OK, so after the experiment...
  - ...we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,
  - on each of the 10 benchmark instances
  - of each of the 10 instance sets.
  - We have $6 * 20 * 10 * 10 = 12\,000$ log files!



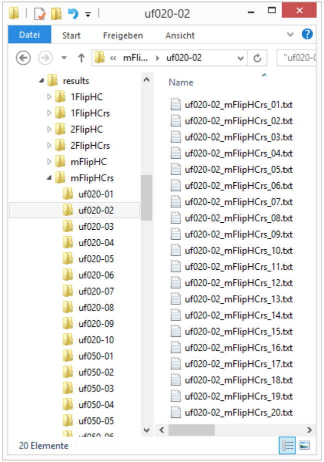
Obtained Data

- OK, so after the experiment...
  - ...we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,
  - on each of the 10 benchmark instances
  - of each of the 10 instance sets.
  - We have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!

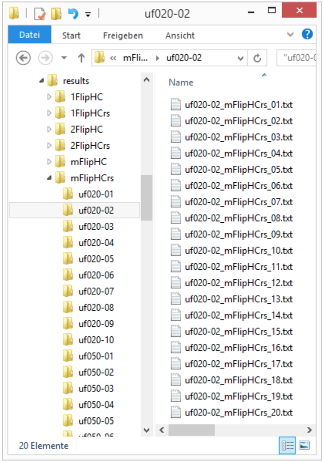


Obtained Data

- OK, so after the experiment we have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!

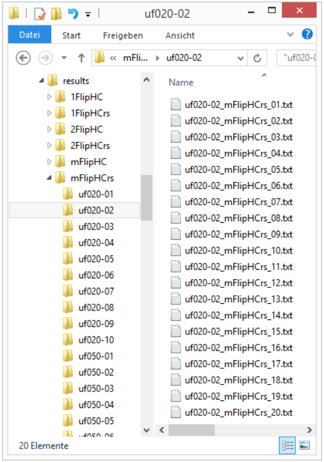


- OK, so after the experiment we have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!
- How can we extract useful information from them



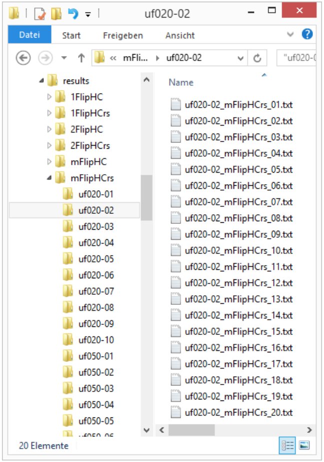
Obtained Data

- OK, so after the experiment we have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!
- How can we extract useful information from them in order to answer the questions which algorithm performs best, when, and why?

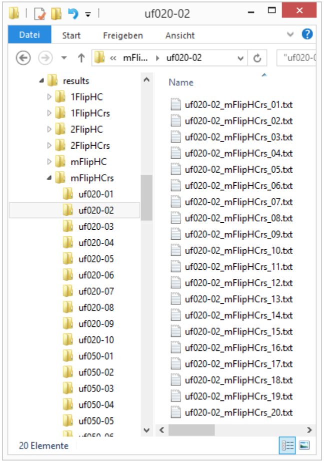


Obtained Data

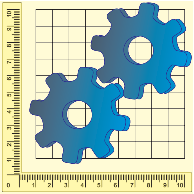
- OK, so after the experiment we have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!
- How can we extract useful information from them in order to answer the questions which algorithm performs best, when, and why?
- What you most likely do: Write your own small program.



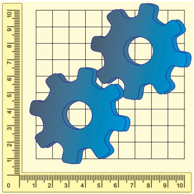
- OK, so after the experiment we have $6 * 20 * 10 * 10 = 12\,000$ log files (with $607\,993$ log points and 8.6 MiB total)!
- How can we extract useful information from them in order to answer the questions which algorithm performs best, when, and why?
- What you most likely do: Write your own small program.
- What you now can do: Use our `optimizationBenchmarking` Evaluator!



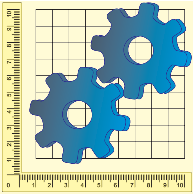
- In the following, I provide some examples for what our evaluator can do.



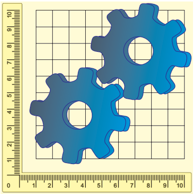
- In the following, I provide some examples for what our evaluator can do.
- First, a quick guide to download and run the example on your computer is given



- In the following, I provide some examples for what our evaluator can do.
- First, a quick guide to download and run the example on your computer is given
- Then, I present some of the evaluation information generated by the Evaluator



- In the following, I provide some examples for what our evaluator can do.
- First, a quick guide to download and run the example on your computer is given
- Then, I present some of the evaluation information generated by the Evaluator
- Finally, I will show *how* that gets done in detail.



- You can quickly download all example data and the Evaluator and run the example on your PC by executing the following code snippet.

Quick Guide

- You can quickly download all example data and the Evaluator and run the example on your PC by executing the following code snippet.
- System Requirements:
  - Linux (for `make.sh`), Windows (for `make.bat`, tested: Win 8, should work also under Win 7)
  - Java 1.7 (ideally a `JDK` under a `JRE` slower and higher memory consumption)
  - `svn`
  - optional: a LaTeX installation, such as TeXLive (needed for generating pdf reports)

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

Listing: Linux: script `make.sh` for downloading & running the MAX-SAT example.

```
#!/bin/bash

jarName="optimizationBenchmarking-full.jar"

outputDir=`pwd`
echo "Writing output to folder '${outputDir}'"

echo "Downloading experimental results via 'svn export' from GitHub."
svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/maxSat/results
echo "Downloading evaluation/configuration via 'svn export' from GitHub."
svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/maxSat/evaluation

jarDownloadURL=$(wget "http://optimizationbenchmarking.github.io/optimizationBenchmarking/currentVersion.url" -q -O -)
echo "Downloading evaluator from '${jarDownloadURL}'."
wget -O "${outputDir}/${jarName}" "${jarDownloadURL}"

echo "Applying evaluator and obtaining reports in different formats."
cd "${outputDir}/evaluation"
java -jar "${outputDir}/${jarName}" -configXML=configForIEEEtran.xml
java -jar "${outputDir}/${jarName}" -configXML=configForLNCS.xml
java -jar "${outputDir}/${jarName}" -configXML=configForSigAlternate.xml
java -jar "${outputDir}/${jarName}" -configXML=configForXHTML.xml
java -jar "${outputDir}/${jarName}" -configXML=configForExport.xml

cd "${outputDir}"
echo "Done."
```

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

Listing: Windows: script `make.bat` for downloading & running the MAX-SAT example.

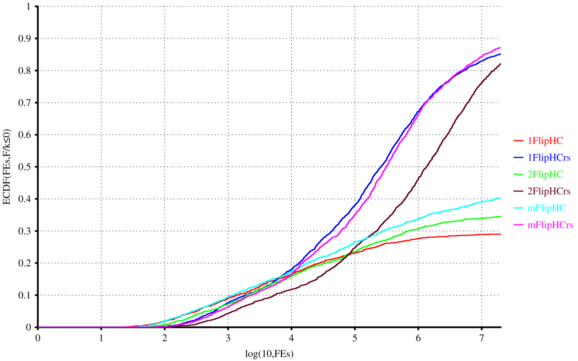
```
echo "Downloading evaluator."
powershell -command "& {iwr http://optimizationbenchmarking.github.io/optimizationBenchmarking/currentVersion.url -OutFile version.txt}"
for /F "delims=" %i in (version.txt) do set downloadURL=%i
powershell -command "& {iwr %downloadURL% -OutFile optimizationBenchmarking.jar}"
del version.txt

echo "Downloading (but not installing!) required 3rd-party software: downloading SVN client and 7-Zip to extract it."
md svn
cd svn
powershell -command "& {iwr https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/raw/master/tools/windows/7zip/7za.exe -OutFile 7za.exe}"
powershell -command "& {iwr https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/raw/master/tools/windows/svn/svn.tar.lzma -OutFile svn.tar.lzma}"
7za x svn.tar.lzma
7za x svn.tar
cd..

echo "Downloading experimental results via 'svn-export' from GitHub."
svn\svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/maxSat/results

echo "Downloading evaluation/configuration via 'svn export' from GitHub."
svn\svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/maxSat/evaluation

rd /s /q svn

echo "Applying evaluator and obtaining reports in different formats."
cd evaluation
java -jar "..\optimizationBenchmarking.jar" -configXML=configForIEEEtran.xml
java -jar "..\optimizationBenchmarking.jar" -configXML=configForLNCS.xml
java -jar "..\optimizationBenchmarking.jar" -configXML=configForSigAlternate.xml
java -jar "..\optimizationBenchmarking.jar" -configXML=configForXHTML.xml
java -jar "..\optimizationBenchmarking.jar" -configXML=configForExport.xml

cd..
echo "Done."
```

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)
- After the script, you will have
  - a folder `results` with the log files which have been evaluated
  - a folder `evaluation` with the configuration files and the `evaluation.xml` file defining what to do
  - a filder `reports` with the generated reports

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)
- After the script, you will have
  - a folder `results` with the log files which have been evaluated
  - a folder `evaluation` with the configuration files and the `evaluation.xml` file defining what to do
  - a filder `reports` with the generated reports
- But now, let's continue with the example…

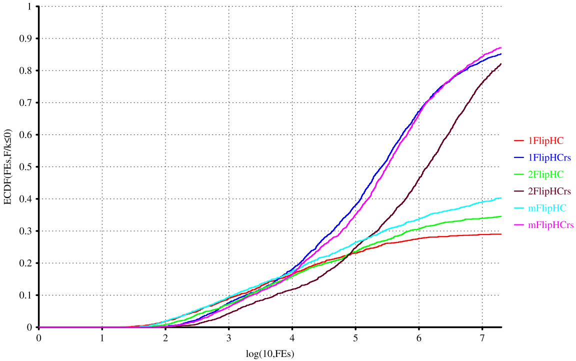
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.

- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



The ECDF in over all 100 benchmark instances for time measure *FEs* (log-scaled).

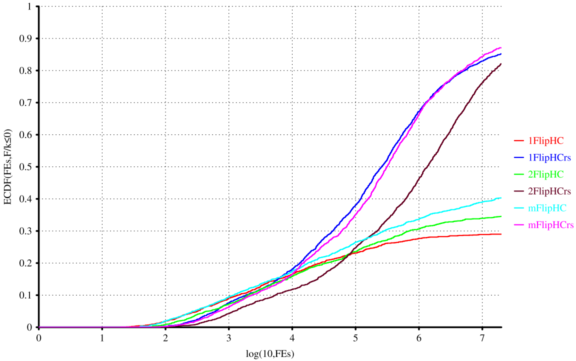
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



The ECDF in over all 100 benchmark instances for time measure *FEs* (log-scaled).

- the methods with restarts solve more problems (up to 90%!)

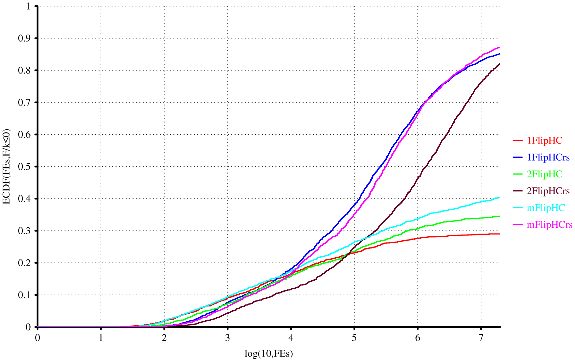
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



The ECDF in over all 100 benchmark instances for time measure *FEs* (log-scaled).

- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips

- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.

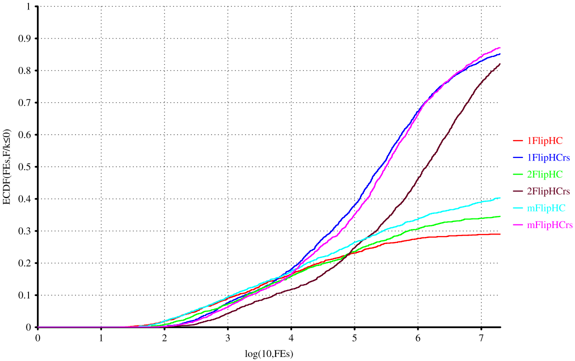


- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips
- oddly, for restart HCers, there is a tie between the m- and 1-flip versions

The ECDF in over all 100 benchmark instances for time measure *FEs* (log-scaled).

University of Science and Technology of China

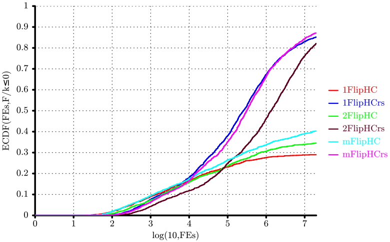
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips
- oddly, for restart HCers, there is a tie between the m- and 1-flip versions

The ECDF in over all 100 benchmark instances for time measure *FEs* (log-scaled, optimized for `IEEEtran` and two figures per row).

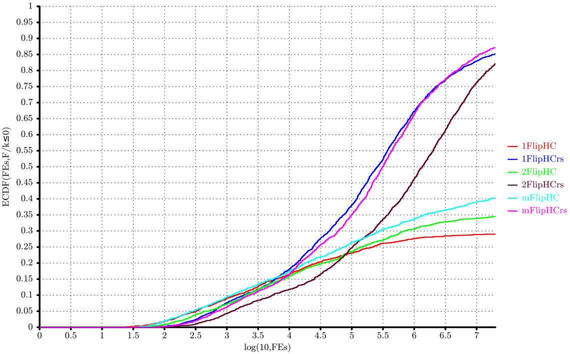
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips
- oddly, for restart HCers, there is a tie between the m- and 1-flip versions

The ECDF in over all 100 benchmark instances (log-scaled, optimized for LNCS and two figures per row).

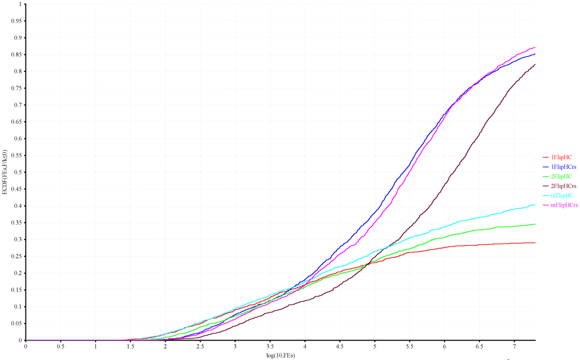
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips
- oddly, for restart HCers, there is a tie between the m- and 1-flip versions

The ECDF in over all 100 benchmark instances (log-scaled, optimized for sig-alternate and two figures per row).

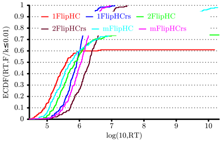
- We can plot the Empirical (Cumulative) Distribution Function (ECDF) [66, 72, 80, 84] for us, which provides the fraction of runs that have found the solution for their respective problem at a given point in time.



The ECDF in over all 100 benchmark instances (log-scaled, optimized for XHTML and two figures per row).

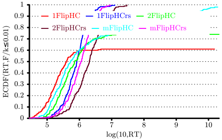
- the methods with restarts solve more problems (up to 90%!)
- plain m-flips are better than 2-flips are better than 1-flips
- oddly, for restart HCers, there is a tie between the m- and 1-flip versions

We now look at the ECDF for different values of n and a goal of 1% unsatisfied clauses over RT (log-scaled).

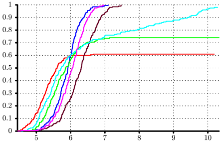


legend

For $n = 20$, the methods with restarts are better.



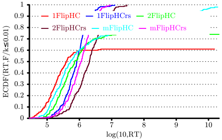
legend



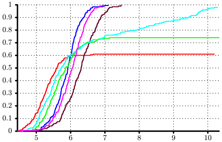
$n = 20$

University of Science and Technology of China

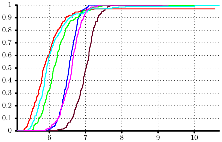
But for $n \geq 50$, those without reach the goal faster.



legend



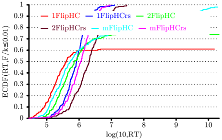
$n = 20$



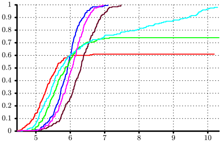
$n = 50$

ECDF for Different Values of n

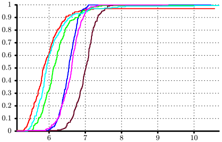
It seems that 1% unsatisfied clauses can be reached with 1-flips and without restarts.



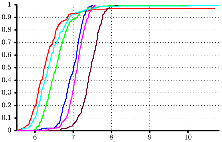
legend



$n = 20$



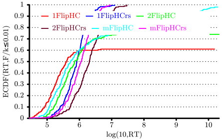
$n = 50$



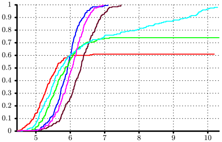
$n = 75$

ECDF for Different Values of n

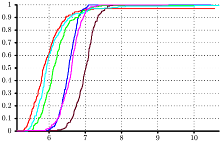
The 2-flip operator again performs worst.



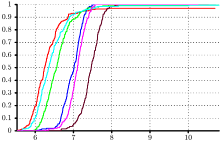
legend



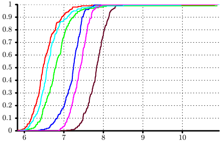
$n = 20$



$n = 50$



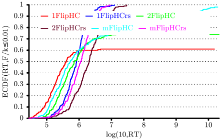
$n = 75$



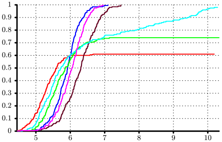
$n = 100$

ECDF for Different Values of n

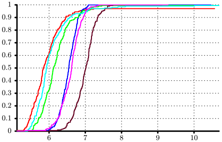
It looks as if it gets easier to attain a 1% error margin if n increases (all ECDFs reach 1).



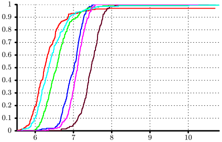
legend



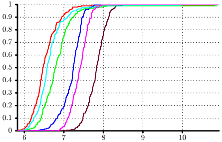
$n = 20$



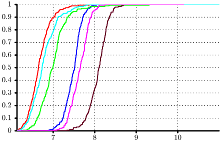
$n = 50$



$n = 75$



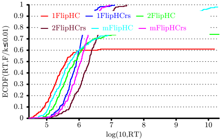
$n = 100$



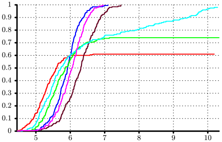
$n = 125$

ECDF for Different Values of n

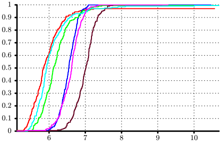
For small problems, 1-flip is slightly faster than m-flip.



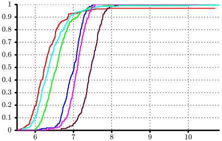
legend



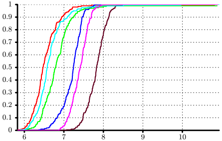
$n = 20$



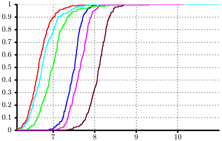
$n = 50$



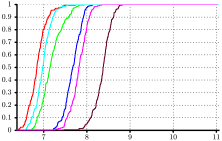
$n = 75$



$n = 100$



$n = 125$

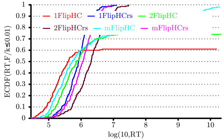


$n = 150$

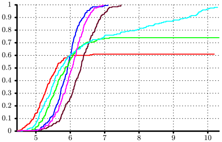
ECDF for Different Values of n

中国科学技术大学
University of Science and Technology of China

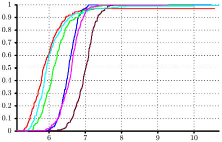
For small problems, 1-flip is slightly faster than m-flip.



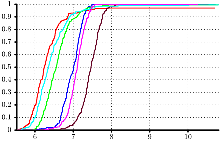
legend



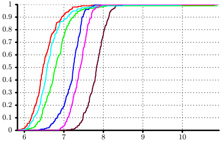
$n = 20$



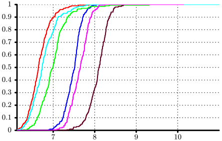
$n = 50$



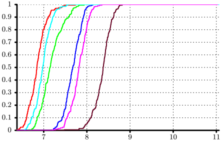
$n = 75$



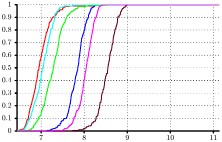
$n = 100$



$n = 125$



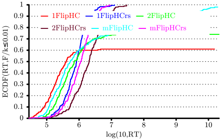
$n = 150$



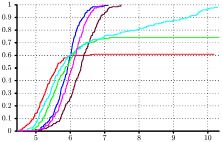
$n = 175$

ECDF for Different Values of n

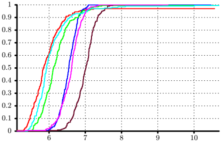
For larger problems, m-flip becomes slightly faster.



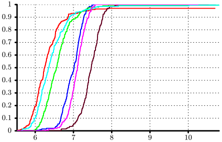
legend



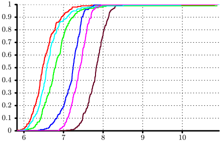
$n = 20$



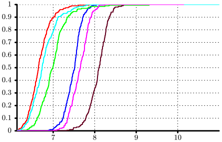
$n = 50$



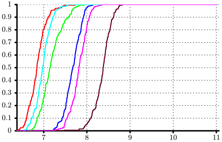
$n = 75$



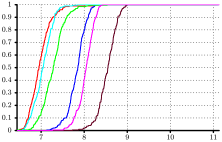
$n = 100$



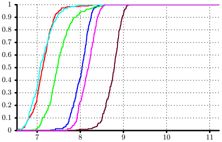
$n = 125$



$n = 150$



$n = 175$

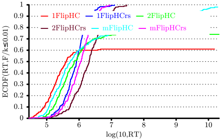


$n = 200$

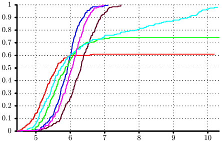
ECDF for Different Values of n

中国科学技术大学
University of Science and Technology of China

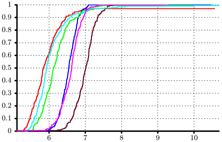
All in all, similar behavior over all scales (reaching 1% error seems to be easy).



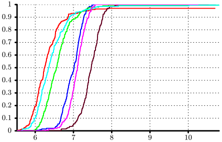
legend



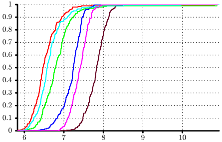
$n = 20$



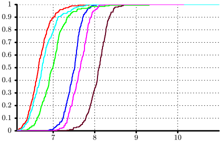
$n = 50$



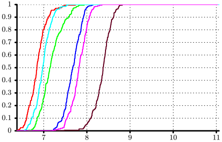
$n = 75$



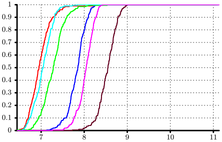
$n = 100$



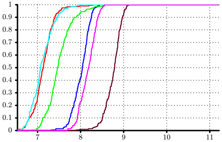
$n = 125$



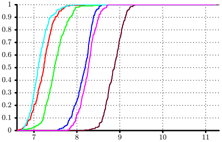
$n = 150$



$n = 175$



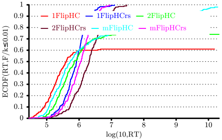
$n = 200$



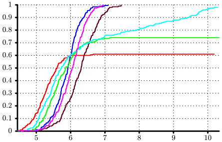
$n = 225$

ECDF for Different Values of n

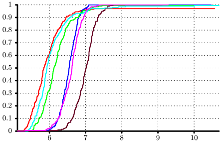
Only required runtime increases by up to 100 times.



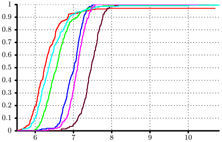
legend



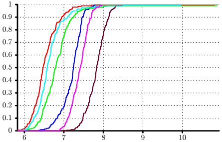
$n = 20$



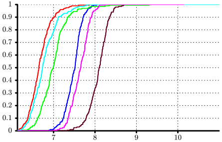
$n = 50$



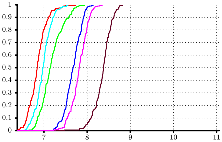
$n = 75$



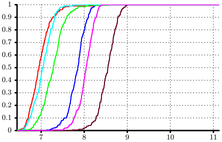
$n = 100$



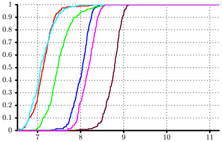
$n = 125$



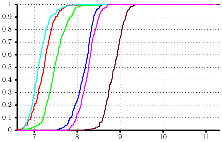
$n = 150$



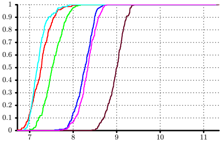
$n = 175$



$n = 200$

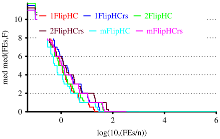


$n = 225$



$n = 250$

We now look at the progress curves (F over *FEs* divided by[1] n, log-scaled) for different values of k.



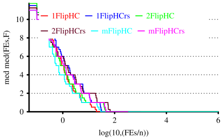
legend

[1]We normalize *FEs* with n in the hope to make the time measure comparable over different n.

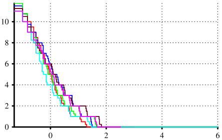
Progress for Different Values of $\mathbf{k}$

University of Science and Technology of China

For very small-scale problems, all algorithms behave similar.

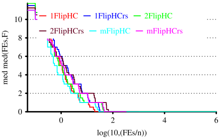


legend

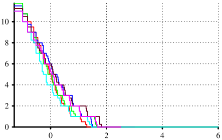


$k = 91$

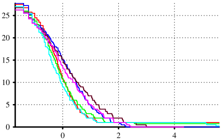
But soon, two groups form: with and without restarts.



legend

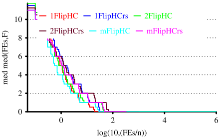


$k = 91$

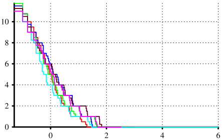


$k = 218$

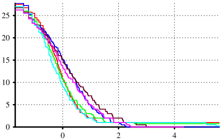
Algorithms using *my example restart policy* seem to be slower.



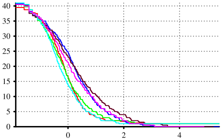
legend



$k = 91$



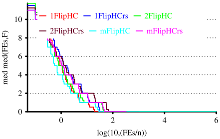
$k = 218$



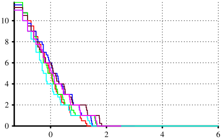
$k = 325$

Progress for Different Values of k

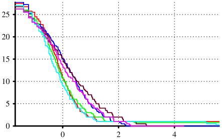
The gap increases with rising k



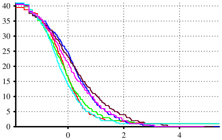
legend



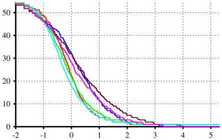
$k = 91$



$k = 218$

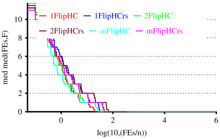


$k = 325$

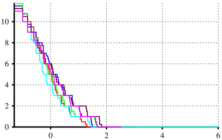


$k = 430$

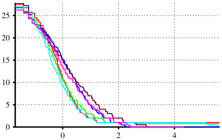
Thus, we find: algorithms with my restart policy are slower than those without…



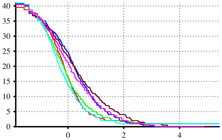
legend



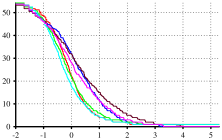
$k = 91$



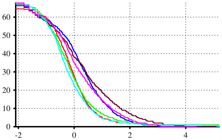
$k = 218$



$k = 325$

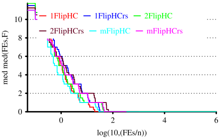


$k = 430$

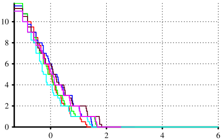


$k = 538$

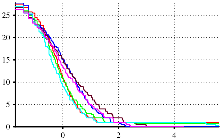
. . . but from the ECDF we know they can solve more problems eventually.



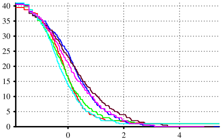
legend



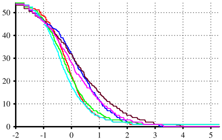
$k = 91$



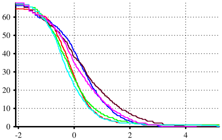
$k = 218$



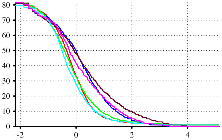
$k = 325$



$k = 430$



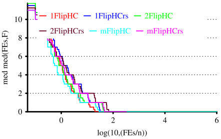
$k = 538$



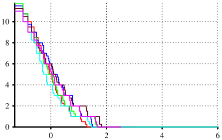
$k = 645$

Progress for Different Values of $\mathbf{k}$

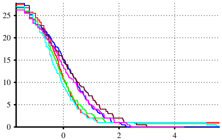
For all scales, the initial random solutions, seem to have about 12% of unsatisfied clauses (in median).



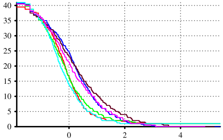
legend



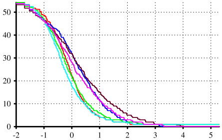
$k = 91$



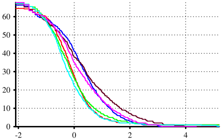
$k = 218$



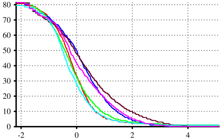
$k = 325$



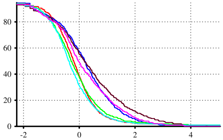
$k = 430$



$k = 538$



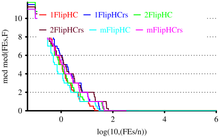
$k = 645$



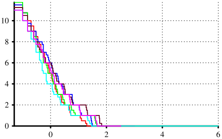
$k = 753$

Progress for Different Values of k

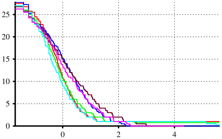
Convergence seems to happen between $100n$ and $1000n$



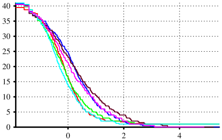
legend



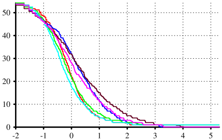
$k = 91$



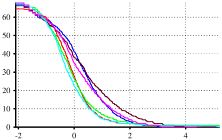
$k = 218$



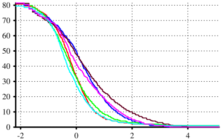
$k = 325$



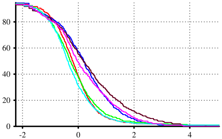
$k = 430$



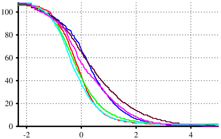
$k = 538$



$k = 645$



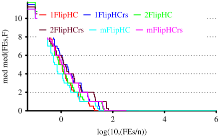
$k = 753$



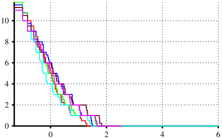
$k = 860$

Progress for Different Values of k

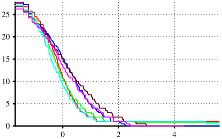
Convergence seems to happen between $100n$ and $1000n$



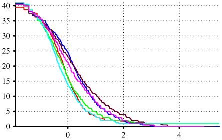
legend



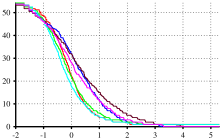
$k = 91$



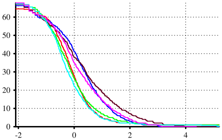
$k = 218$



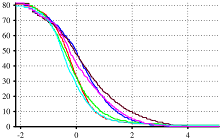
$k = 325$



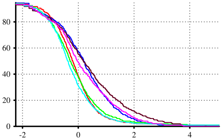
$k = 430$



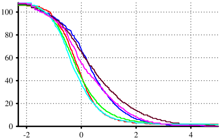
$k = 538$



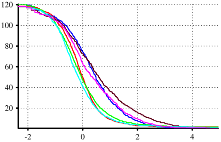
$k = 645$



$k = 753$



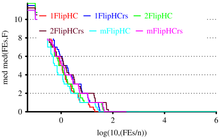
$k = 860$



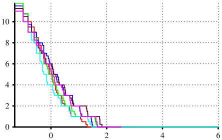
$k = 960$

Progress for Different Values of k

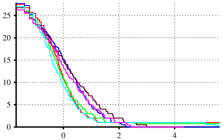
Convergence seems to happen between $100n$ and $1000n$



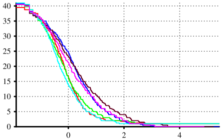
legend



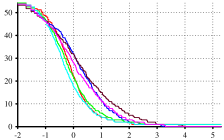
$k = 91$



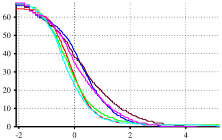
$k = 218$



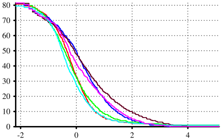
$k = 325$



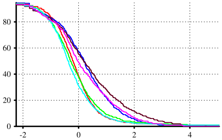
$k = 430$



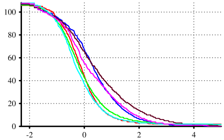
$k = 538$



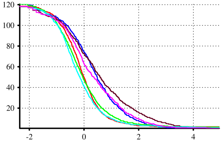
$k = 645$



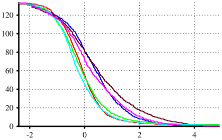
$k = 753$



$k = 860$

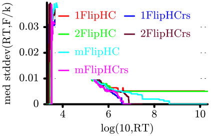


$k = 960$



$k = 1065$

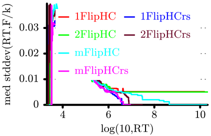
Let's look at the standard deviation of the best objective value F (divided by[1] k) found over RT (log-scaled) for different values of n.



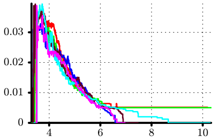
legend

[1] Since F is always in $1 \dots k$, dividing it by k normalizes it into $[0, 1]$ and makes the values comparable for different k or n.

For small-scale problems, the standard deviation seems to decrease steadily.

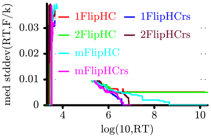


legend

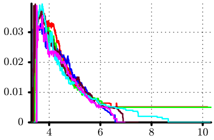


$n = 20$

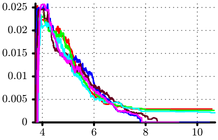
The reason is probably that the algorithms converge nicely.



legend

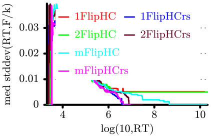


$n = 20$

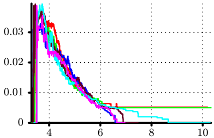


$n = 50$

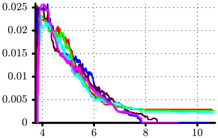
For the methods with restarts, it reaches very close to 0.



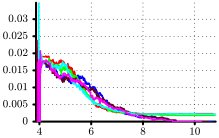
legend



$n = 20$

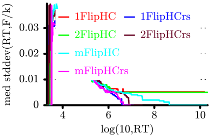


$n = 50$

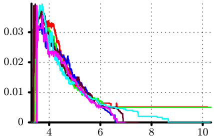


$n = 75$

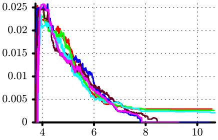
For those without, it remains constant above 0 after some time.



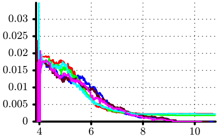
legend



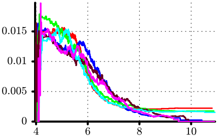
$n = 20$



$n = 50$



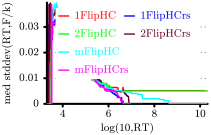
$n = 75$



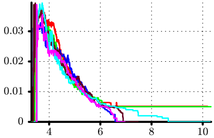
$n = 100$

StdDev of F for Different Values of n

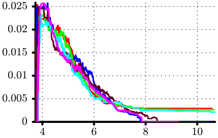
These algorithms probably get stuck at different local optima in different runs.



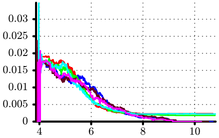
legend



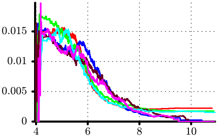
$n = 20$



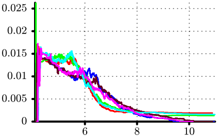
$n = 50$



$n = 75$

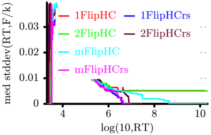


$n = 100$

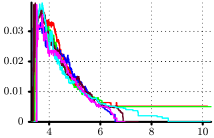


$n = 125$

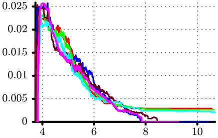
For increasing scales, the standard deviation goes first down, then up, then farther down.



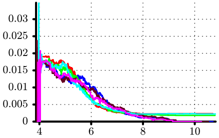
legend



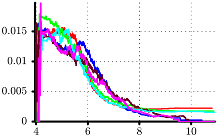
$n = 20$



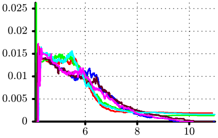
$n = 50$



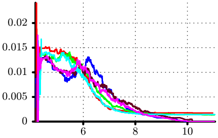
$n = 75$



$n = 100$



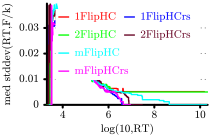
$n = 125$



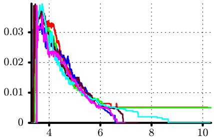
$n = 150$

StdDev of F for Different Values of $\mathbf{n}$

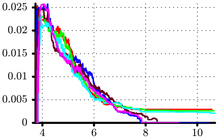
Maybe there is some kind of hard-to-attain improvement that some runs find earlier than others.



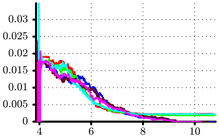
legend



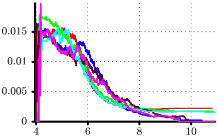
$n = 20$



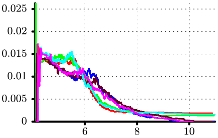
$n = 50$



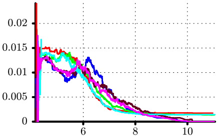
$n = 75$



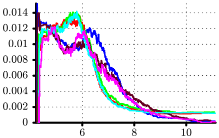
$n = 100$



$n = 125$



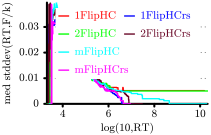
$n = 150$



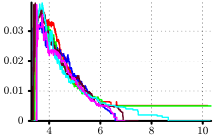
$n = 175$

StdDev of F for Different Values of n

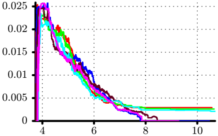
The time of convergence seems to increase for the methods with restarts with n.



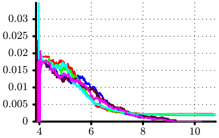
legend



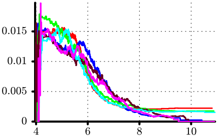
$n = 20$



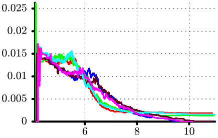
$n = 50$



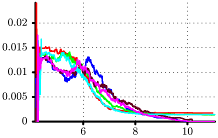
$n = 75$



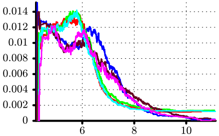
$n = 100$



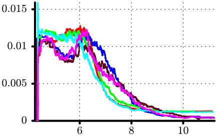
$n = 125$



$n = 150$



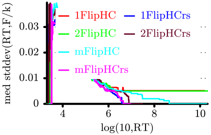
$n = 175$



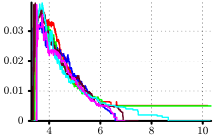
$n = 200$

StdDev of F for Different Values of $\mathbf{n}$

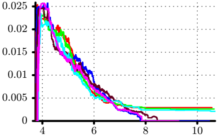
The early standard deviations are usually below 0.03 and highest for small n.



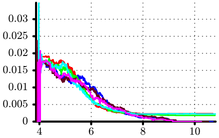
legend



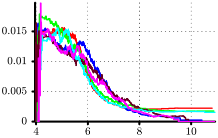
$n = 20$



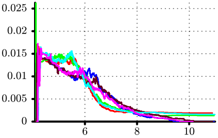
$n = 50$



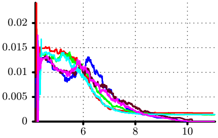
$n = 75$



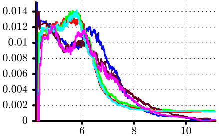
$n = 100$



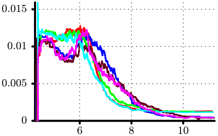
$n = 125$



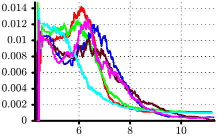
$n = 150$



$n = 175$



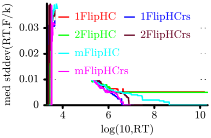
$n = 200$



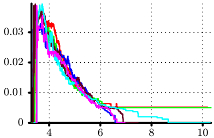
$n = 225$

StdDev of F for Different Values of n

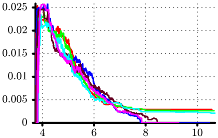
The early standard deviations are usually below 0.03 and highest for small n.



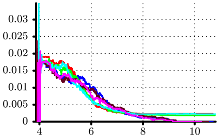
legend



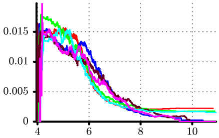
$n = 20$



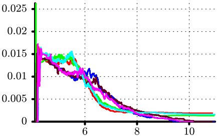
$n = 50$



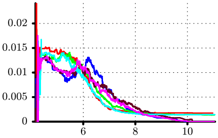
$n = 75$



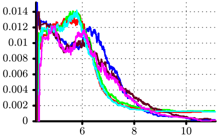
$n = 100$



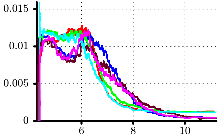
$n = 125$



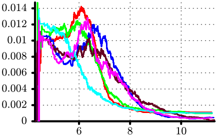
$n = 150$



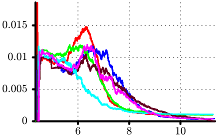
$n = 175$



$n = 200$

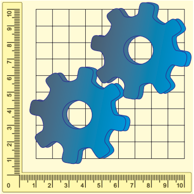


$n = 225$

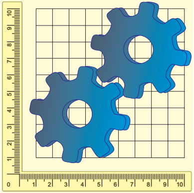


$n = 250$

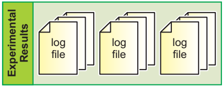
- So these are *some* of the things `optimizationBenchmarking` can *currently* do.

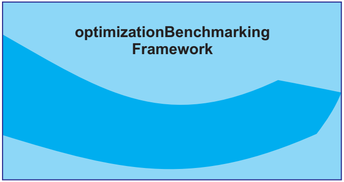


- So these are *some* of the things `optimizationBenchmarking` can *currently* do.
- But how to do them?

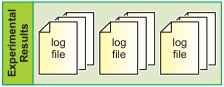


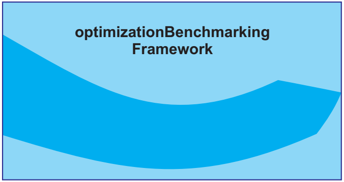
- Let us now take a closer look on how the `optimizationBenchmarking` evaluator is used (and works)



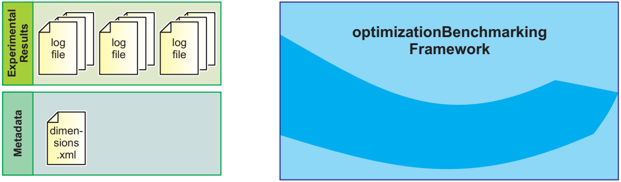


- We got a couple of log files for each experiment

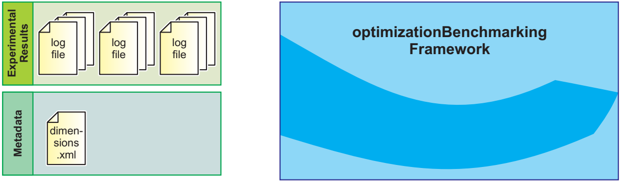




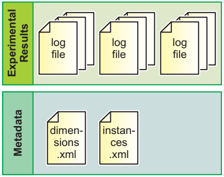
- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files

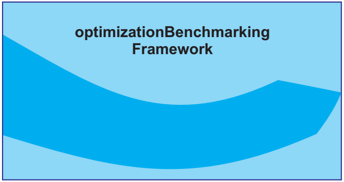


- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured

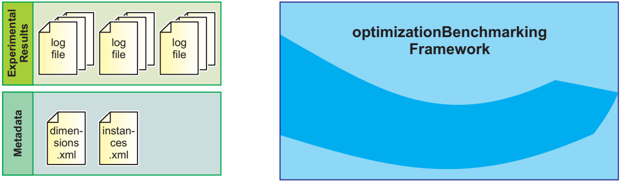


- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured: *FEs*, *RT*, and F in our example

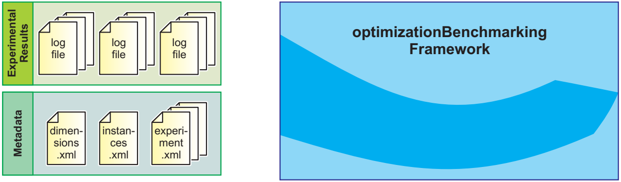




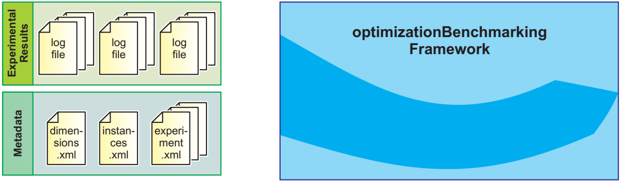
- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured: *FEs*, *RT*, and F in our example
- We specify which benchmark instances we have and what their features are



- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured: *FEs*, *RT*, and F in our example
- We specify which benchmark instances we have and what their features are: 10×10 instances in our example, with features n and k

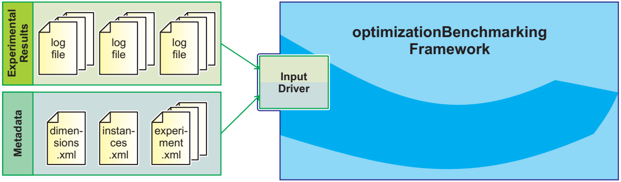


- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured: *FEs*, *RT*, and F in our example
- We specify which benchmark instances we have and what their features are: 10×10 instances in our example, with features n and k
- For each experiment, we specify the parameters

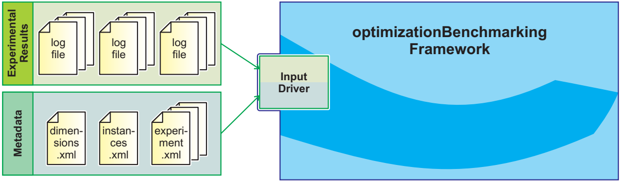


- We got a couple of log files for each experiment: 6 experiments in our example, each with $10 \times 10 \times 20 = 2000$ log files
- We specify which dimensions we have measured: *FEs*, *RT*, and F in our example
- We specify which benchmark instances we have and what their features are: 10×10 instances in our example, with features n and k
- For each experiment, we specify the parameters: in our example, these are `algorithm`, `operator`, `restart`

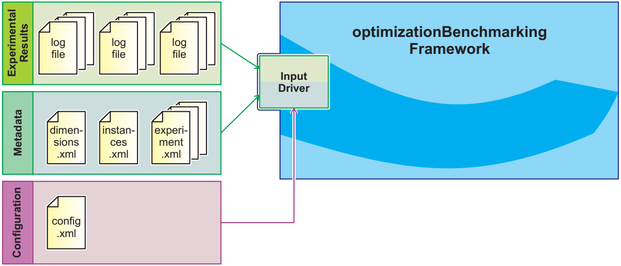
The Flow



- An "input driver" loads the data

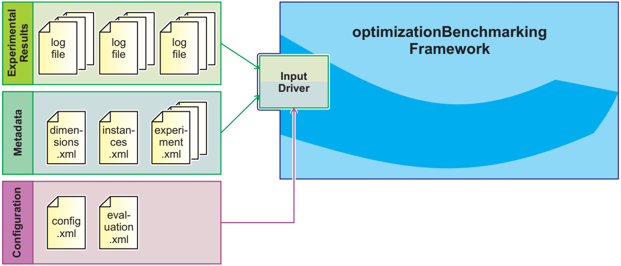


- An "input driver" loads the data: most commonly, the data will be in `CSV+EDI` format, but we also support *BBOB* [71, 80–82], *TSP Suite* [72, 83], and pure `EDI`



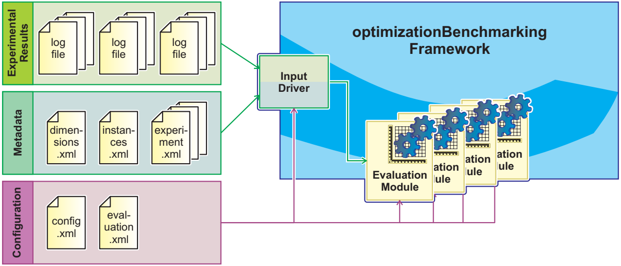
- An "input driver" loads the data: most commonly, the data will be in `CSV+EDI` format, but we also support *BBOB* [71, 80–82], *TSP Suite* [72, 83], and pure `EDI`
- Via a configuration file, we choose which input and output formats to use, as well as which file specifies the evaluation process

The Flow

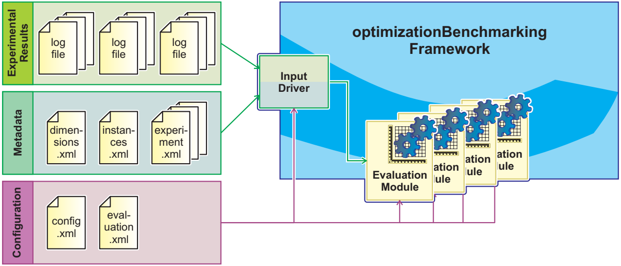


- An "input driver" loads the data: most commonly, the data will be in `CSV+EDI` format, but we also support *BBOB* [71, 80–82], *TSP Suite* [72, 83], and pure `EDI`
- Via a configuration file, we choose which input and output formats to use, as well as which file specifies the evaluation process
- The `evaluation.xml` specifies *how* to evaluate the data, i.e., which evaluation modules to apply

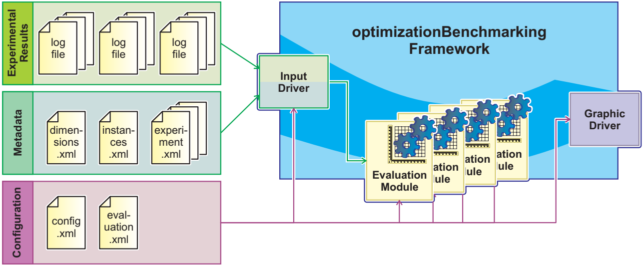
The Flow



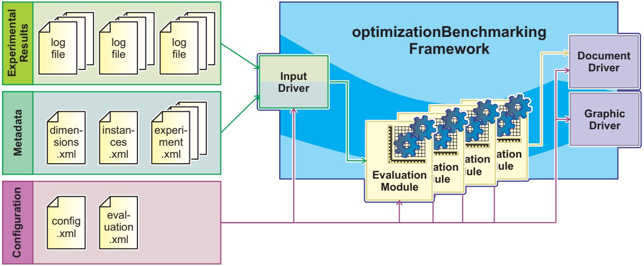
- An evaluation module prints on particular type of information about an experiment or experiment set, such as the ECDF, or a table with final results, etc. . .



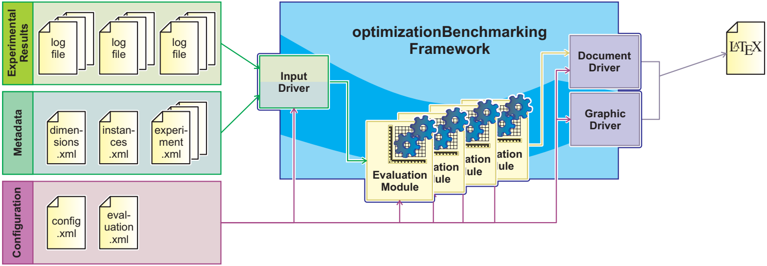
- An evaluation module prints on particular type of information about an experiment or experiment set, such as the ECDF, or a table with final results, etc. . .
- Evaluation modules can be applied multiple times, with different configurations (e.g., we can plot ECDFs for different target solution qualities)



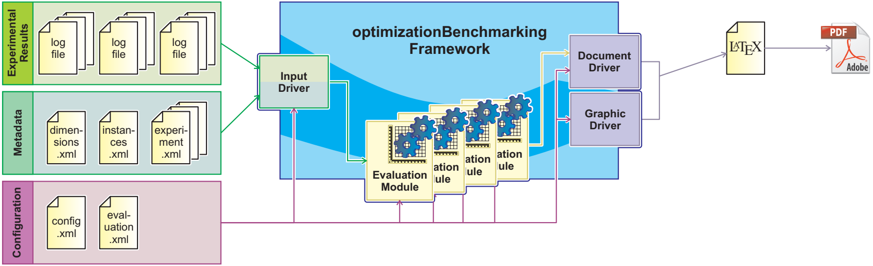
- We can choose among several different formats to be used for graphics, including EPS [85], PDF [86], PGF (LaTeX), SVG(Z), EMF, PNG [87], GIF [88], BMP, and JPG



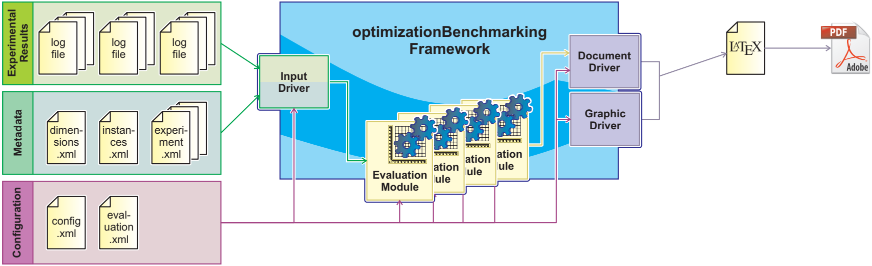
- We can also choose among different formats for the report documents, including. . .



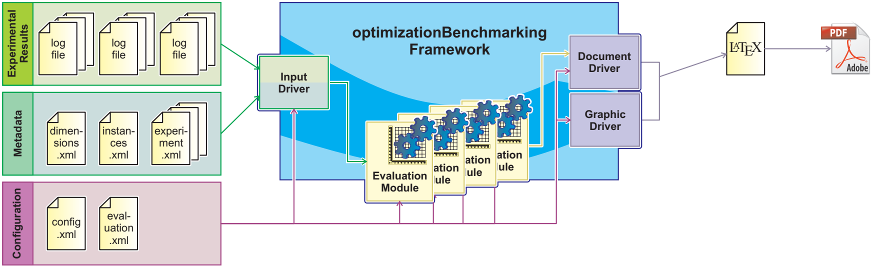
- We can also choose among different formats for the report documents, including LaTeX [89–92]



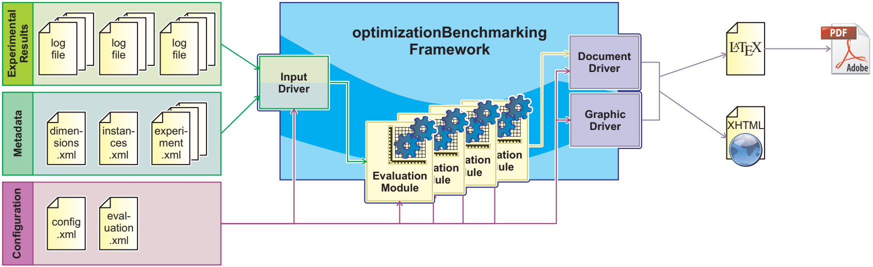
- We can also choose among different formats for the report documents, including LaTeX [89–92]:
  - can automatically be compiled to PDF [86], if a LaTeX compiler (such as TeXLive [93] or MiKTeX [94]) is auto-detected



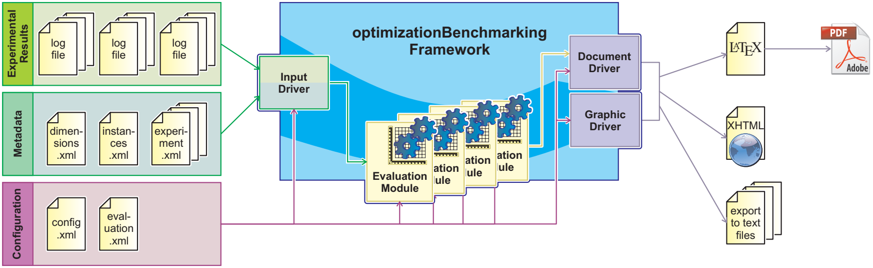
- We can also choose among different formats for the report documents, including LaTeX [89–92]:
  - can automatically be compiled to PDF [86], if a LaTeX compiler (such as TeXLive [93] or MiKTeX [94]) is auto-detected
  - different document classes, such as IEEEtran [95], Springer LLNCS [96], ACM sig-alternate [97] can be chosen



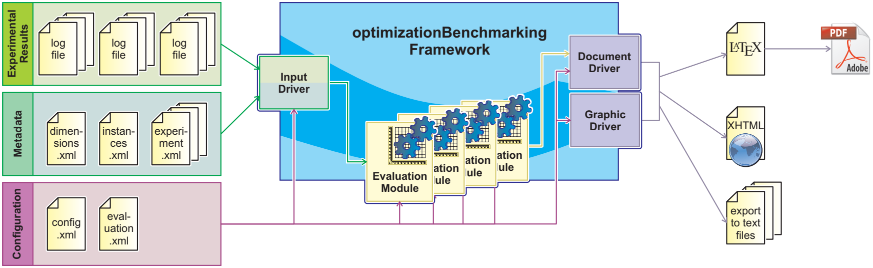
- We can also choose among different formats for the report documents, including LaTeX [89–92]:
  - can automatically be compiled to PDF [86], if a LaTeX compiler (such as TeXLive [93] or MiKTeX [94]) is auto-detected
  - different document classes, such as IEEEtran [95], Springer LLNCS [96], ACM sig-alternate [97] can be chosen
  - graphic sizes and fonts used in graphics are automatically adapted to document class



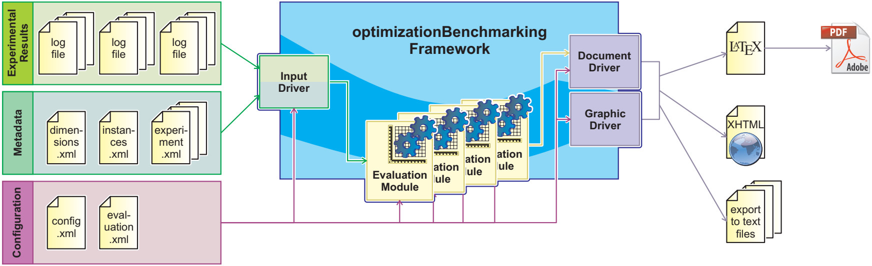
- We can also choose among different formats for the report documents, including LaTeX and XHTML [98] for quick viewing in a browser



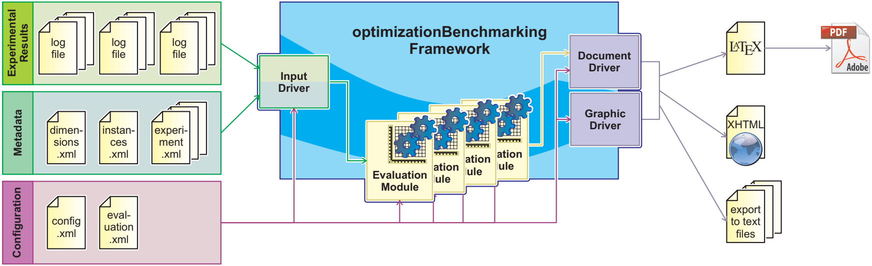
- We can also choose among different formats for the report documents, including LaTeX, XHTML [98], and a plain text format to export results to other applications



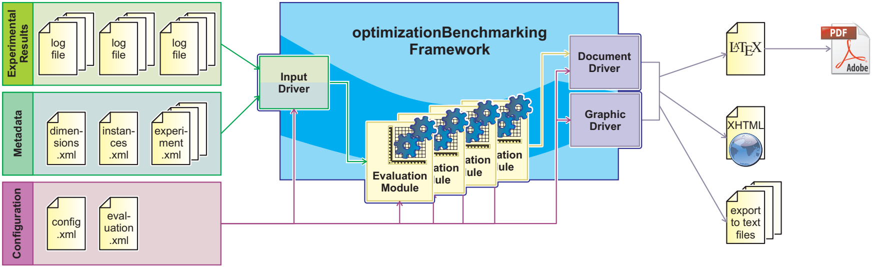
- We can also choose among different formats for the report documents, including LaTeX, XHTML [98], and a plain text format to export results to other applications
- Evaluation Modules as well as Input, Document, and Graphic Drivers can easily be added



- We can also choose among different formats for the report documents, including LaTeX, XHTML [98], and a plain text format to export results to other applications
- Evaluation Modules as well as Input, Document, and Graphic Drivers can easily be added: implement the corresponding interface



- We can also choose among different formats for the report documents, including LaTeX, XHTML [98], and a plain text format to export results to other applications
- Evaluation Modules as well as Input, Document, and Graphic Drivers can easily be added: implement the corresponding interface, throw your class into the classpath



- We can also choose among different formats for the report documents, including LaTeX, XHTML [98], and a plain text format to export results to other applications
- Evaluation Modules as well as Input, Document, and Graphic Drivers can easily be added: implement the corresponding interface, throw your class into the classpath, and tell the system to use it in the `config.xml` or `evaluation.xml`...

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)
    - `iterationSubFE`, a finer-grained machine independent measure, e.g., bit flips in SAT problems [66], distance evaluations in TSP [72]

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)
    - `iterationSubFE`, a finer-grained machine independent measure, e.g., bit flips in SAT problems [66], distance evaluations in TSP [72]
    - `runtimeCPU`, i.e., processor time (machine dependent)

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)
    - `iterationSubFE`, a finer-grained machine independent measure, e.g., bit flips in SAT problems [66], distance evaluations in TSP [72]
    - `runtimeCPU`, i.e., processor time (machine dependent)
    - `runtimeNormalized`, a machine-independent time measure, maybe `runtimeCPU` divide by a performance factor

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)
    - `iterationSubFE`, a finer-grained machine independent measure, e.g., bit flips in SAT problems [66], distance evaluations in TSP [72]
    - `runtimeCPU`, i.e., processor time (machine dependent)
    - `runtimeNormalized`, a machine-independent time measure, maybe `runtimeCPU` divide by a performance factor
    - `qualityProblemDependent` a problem-instance specific objective value (e.g., number of unsatisfied clauses in SAT)

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type, which is either
    - `iterationAlgorithmStep`, e.g., a generation in an EA (machine independent)
    - `iterationFE`, a function evaluation, i.e., a fully constructed candidate solution has been evaluated (machine independent)
    - `iterationSubFE`, a finer-grained machine independent measure, e.g., bit flips in SAT problems [66], distance evaluations in TSP [72]
    - `runtimeCPU`, i.e., processor time (machine dependent)
    - `runtimeNormalized`, a machine-independent time measure, maybe `runtimeCPU` divide by a performance factor
    - `qualityProblemDependent` a problem-instance specific objective value (e.g., *number* of unsatisfied clauses in SAT)
    - `qualityProblemIndependent` an objective value which can compared over different instances (e.g., the *fraction* of unsatisfied clauses in SAT)

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction, which is either
    - `decreasing`, i.e., values get smaller, but consecutive log points may have same value

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction, which is either
    - `decreasing`, i.e., values get smaller, but consecutive log points may have same value
    - `decreasingStrictly`, such as the objective value in the log points of our MAX-SAT example

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction, which is either
    - `decreasing`, i.e., values get smaller, but consecutive log points may have same value
    - `decreasingStrictly`, such as the objective value in the log points of our MAX-SAT example
    - `increasing`, like the absolute runtime: due to clock resolution, some log points may be taken at the same clock time

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction, which is either
    - `decreasing`, i.e., values get smaller, but consecutive log points may have same value
    - `decreasingStrictly`, such as the objective value in the log points of our MAX-SAT example
    - `increasing`, like the absolute runtime: due to clock resolution, some log points may be taken at the same clock time
    - `increasingStrictly`, like the *FEs* in our example – no two log points can have the same value in this dimension

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`,
    - `short`

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`,
    - `short`,
    - `int`

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`,
    - `short`,
    - `int`,
    - `long`

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`,
    - `short`,
    - `int`,
    - `long`,
    - `float`

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type, which is either
    - `byte`,
    - `short`,
    - `int`,
    - `long`,
    - `float`, or
    - `double`

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks, such as
    - `iLowerBound`, a integer lower bound, such as 1 for *FEs*

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks, such as
    - `iLowerBound`, a integer lower bound, such as 1 for *FEs or*
    - `fLowerBound`, a floating point lower bound

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks, such as
    - `iLowerBound`, a integer lower bound, such as 1 for *FEs or*
    - `fLowerBound`, a floating point lower bound
    - `iUpperBound`, a integer upper bound

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks, such as
    - `iLowerBound`, a integer lower bound, such as 1 for *FEs or*
    - `fLowerBound`, a floating point lower bound
    - `iUpperBound`, a integer upper bound *or*
    - `fUpperBound`, a floating point upper bound

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks, and
  - an optional description

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks,
  - an optional description
- With this information, the nature of measurements is defined and data can be validated

Measured Dimensions

- For each research subject, we may collect different “kinds” of measurements
- Each such “kind” corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks,
  - an optional description
- With this information, the nature of measurements is defined and data can be validated
- Multiple time and quality dimensions can be specified

Measured Dimensions

- For each research subject, we may collect different "kinds" of measurements
- Each such "kind" corresponds to one *dimension*
- A dimension has
  - a name,
  - a type,
  - a direction,
  - a data type,
  - bounds which can be used in computations and for sanity checks,
  - an optional description
- With this information, the nature of measurements is defined and data can be validated
- Multiple time and quality dimensions can be specified
- Diagrams can be plotted and values can be analyized according to different dimensions

中国科学技术大学 University of Science and Technology of China

- To specify all this, we can make an XML file called dimensions.xml and put it into the results folder with our log files.

Listing: File dimensions.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<dimensions
  xmlns="http://www.optimizationBenchmarking.org/formats/
      experimentDataInterchange/experimentDataInterchange.1.0.xsd">

  <dimension name="FEs"
    description="The number of function evaluations, i.e., the amount of
       generated candidate solutions."
    dimensionType="iterationFE" direction="increasingStrictly" dataType="long"
    iLowerBound="1" />

  <dimension name="RT" description="The elapsed runtime in nanoseconds."
    dimensionType="runtimeCPU" direction="increasing" dataType="long"
    iLowerBound="0" />

  <dimension name="F" description="The number of unsatisfied clauses."
    dimensionType="qualityProblemDependent" direction="decreasing"
    dataType="int" iLowerBound="0" iUpperBound="2000" />

</dimensions>
```

- In an experiment, an optimization algorithm is applied to different *benchmark instances*

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example
  - each feature has
    - a name (such as n)

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example
  - each feature has
    - a name (such as n),
    - a value (such as 250)

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example
  - each feature has
    - a name (such as n),
    - a value (such as 250),
    - an optional description

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example
  - each feature has
    - a name (such as n),
    - a value (such as 250),
    - an optional description, and
    - an optional value description

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension
    - makes particular sense for `qualityProblemDependent`

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension
    - makes particular sense for `qualityProblemDependent`
    - specified as element `bounds` with attribute `dimension` and either `iLowerBound` or `fLowerBound` and/or either `iUpperBound` or `fUpperBound`

Benchmark Instances

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description
- Feature specifications allow us to explore relationship between instance features and algorithm behavior

Benchmark Instances

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description
- Feature specifications allow us to explore relationship between instance features and algorithm behavior
- Any number of features can be defined, but all instances much specify the same features (may with different values)

Benchmark Instances

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description
- Feature specifications allow us to explore relationship between instance features and algorithm behavior
- Any number of features can be defined, but all instances much specify the same features (may with different values)
- Any feature value type is possible, numerical features are automatically detected

Benchmark Instances

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description
- Feature specifications allow us to explore relationship between instance features and algorithm behavior
- Any number of features can be defined, but all instances much specify the same features (may with different values)
- Any feature value type is possible, numerical features are automatically detected
- Numerical features can be used in formulas and computations, e.g., to normalize values

Benchmark Instances

- In an experiment, an optimization algorithm is applied to different *benchmark instances*
- Each instance has
  - a name,
  - *features*, such as n or k in our example,
  - optional bounds for each dimension, and
  - an optional description
- Feature specifications allow us to explore relationship between instance features and algorithm behavior
- Any number of features can be defined, but all instances much specify the same features (may with different values)
- Any feature value type is possible, numerical features are automatically detected
- Numerical features can be used in formulas and computations, e.g., to normalize values
- Bounds allow us to validate measured data and can be used in computations

- To specify all this, we can make an XML file called `instances.xml` and put it into the `results` folder with our log files.

Listing: Excerpt from file `instances.xml` for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<instances
  xmlns="http://www.optimizationBenchmarking.org/formats/experimentDataInterchange/experimentDataInterchange
      .1.0.xsd">
  <instance name="uf020-01"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 20 variables and 91 clauses.
    <feature name="n" value="20" />
    <feature name="k" value="91" />
  </instance>
  <instance name="uf020-02"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 20 variables and 91 clauses.
    <feature name="n" value="20" />
    <feature name="k" value="91" />
  </instance>
  <instance name="uf075-01"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 75 variables and 325 clauses
    <feature name="n" value="75" />
    <feature name="k" value="325" />
  </instance>
  <instance name="uf075-02"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 75 variables and 325 clauses
    <feature name="n" value="75" />
    <feature name="k" value="325" />
  </instance>
```

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example
  - each parameter has
    - a name (such as "operator")

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example
  - each parameter has
    - a name (such as "operator"),
    - a value (such as "2-flip")

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example
  - each parameter has
    - a name (such as "operator"),
    - a value (such as "2-flip"),
    - an optional description

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example
  - each parameter has
    - a name (such as "operator"),
    - a value (such as "2-flip"),
    - an optional description, and
    - an optional value description

Experiments

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example,
  - an optional description

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example,
  - an optional description
- Parameter specifications allow us to explore the relationship of parameter settings and algorithm performance
- The algorithm itself is treated as parameter as well

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example,
  - an optional description
- Parameter specifications allow us to explore the relationship of parameter settings and algorithm performance
- The algorithm itself is treated as parameter as well
- Any number of parameters can be defined, different experiments may specify different parameters (e.g., an EA has a population size, HC has not)

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has
  - a name,
  - *parameters*, such as the search operation and whether we do restarts in our example,
  - an optional description
- Parameter specifications allow us to explore the relationship of parameter settings and algorithm performance
- The algorithm itself is treated as parameter as well
- Any number of parameters can be defined, different experiments may specify different parameters (e.g., an EA has a population size, HC has not)
- Any parameter value type is possible, numerical features are automatically detected

- An experiment is the application of an algorithm setup to some (or all) of the benchmark instances, usually for several independent runs on each
- Each experiment has a name, parameters, and an optional description
- Parameter specifications allow us to explore the relationship of parameter settings and algorithm performance
- The algorithm itself is treated as parameter as well
- Any number of parameters can be defined, different experiments may specify different parameters (e.g., an EA has a population size, HC has not)
- Any parameter value type is possible, numerical features are automatically detected
- Numerical parameter values can be used in computations (e.g., to multiply a "generations" dimension of experiments with an EA with the population size

- To specify all this, we can make a separate XML file called experiment.xml for each experiment and put it into root folder of the experiment, e.g., results/1FlipHC.

Listing: Excerpt from file experiment.xml for the 1-flip Hill Climber without restarts.

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment
  xmlns="http://www.optimizationBenchmarking.org/formats/
      experimentDataInterchange/experimentDataInterchange.1.0.xsd"
  name="1FlipHC" description="An experiment with a 1-flip Hill
      Climber without restarts.">
  <parameter name="algorithm" value="HC" />
  <parameter name="operator" value="1-flip" />
  <parameter name="restart" value="false" />
</experiment>
```

- To specify all this, we can make a separate XML file called experiment.xml for each experiment and put it into root folder of the experiment, e.g., results/1FlipHCrs.

Listing: Excerpt from file experiment.xml for the 1-flip Hill Climber with restarts.

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment
  xmlns="http://www.optimizationBenchmarking.org/formats/
      experimentDataInterchange/experimentDataInterchange.1.0.xsd"
  name="1FlipHCrs" description="An experiment with a 1-flip Hill
      Climber with restarts.">
  <parameter name="algorithm" value="HC" />
  <parameter name="operator" value="1-flip" />
  <parameter name="restart" value="true" />
</experiment>
```

University of Science and Technology of China

- To specify all this, we can make a separate XML file called `experiment.xml` for each experiment and put it into root folder of the experiment, e.g., `results/mFlipHCrs`.

Listing: Excerpt from file `experiment.xml` for the m-flip Hill Climber with restarts.

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment
  xmlns="http://www.optimizationBenchmarking.org/formats/
      experimentDataInterchange/experimentDataInterchange.1.0.xsd"
  name="mFlipHCrs" description="An experiment with a m-flip Hill
      Climber with restarts.">
  <parameter name="algorithm" value="HC" />
  <parameter name="operator" value="m-flip" />
  <parameter name="restart" value="true" />
</experiment>
```

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.

Specifying Evaluation Process

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report

Specifying Evaluation Process

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report
- Modules can be configured, e.g., we can tell the "ECDF" module which dimension we want as x-axis

Specifying Evaluation Process

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report
- Modules can be configured, e.g., we can tell the "ECDF" module which dimension we want as x-axis
- A module can be applied multiple times with different configurations

Specifying Evaluation Process

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report
- Modules can be configured, e.g., we can tell the "ECDF" module which dimension we want as x-axis
- A module can be applied multiple times with different configurations
- A global basic configuration can be provided

Specifying Evaluation Process

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report
- Modules can be configured, e.g., we can tell the "ECDF" module which dimension we want as x-axis
- A module can be applied multiple times with different configurations
- A global basic configuration can be provided
- To specify all this, we supply an XML file called `evaluation.xml`

- Now that we have specified what kind of data we have, we need to tell *what to do with them*.
- The evaluation process of `optimizationBenchmarking` is based on *modules*
- Each module contributes performs one specific computation and adds text and/or figures to the report
- Modules can be configured, e.g., we can tell the "ECDF" module which dimension we want as x-axis
- A module can be applied multiple times with different configurations
- A global basic configuration can be provided
- To specify all this, we supply an XML file called `evaluation.xml`
- In `evaluation.xml`, we can use the names and values of dimensions, features, and parameters

- Global base configuration

Listing: Part 1 from file evaluation.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<e:evaluation
  xmlns:e="http://www.optimizationBenchmarking.org/formats/
      evaluationConfiguration/evaluationConfiguration.1.0.xsd"
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/configuration/
      configuration.1.0.xsd">

  <cfg:configuration>
    <cfg:parameter name="figureSize" value="2 per row" />
    <cfg:parameter name="makeLegendFigure" value="true" />
    <cfg:parameter name="nGrouping" value="distinct" />
    <cfg:parameter name="kGrouping" value="distinct" />
  </cfg:configuration>

  <e:module class="description.instances.InstanceInformation" />
```

Specifying Evaluation Process: evaluation.xml

University of Science and Technology of China

- Global base configuration: 2 figures per row

Listing: Part 1 from file evaluation.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<e:evaluation
  xmlns:e="http://www.optimizationBenchmarking.org/formats/
      evaluationConfiguration/evaluationConfiguration.1.0.xsd"
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/configuration/
      configuration.1.0.xsd">

  <cfg:configuration>
    <cfg:parameter name="figureSize" value="2 per row" />
    <cfg:parameter name="makeLegendFigure" value="true" />
    <cfg:parameter name="nGrouping" value="distinct" />
    <cfg:parameter name="kGrouping" value="distinct" />
  </cfg:configuration>

  <e:module class="description.instances.InstanceInformation" />
```

- Global base configuration: 2 figures per row, figure series should have dedicated sub-figure for legend

Listing: Part 1 from file evaluation.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<e:evaluation
  xmlns:e="http://www.optimizationBenchmarking.org/formats/
      evaluationConfiguration/evaluationConfiguration.1.0.xsd"
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/configuration/
      configuration.1.0.xsd">

  <cfg:configuration>
    <cfg:parameter name="figureSize" value="2 per row" />
    <cfg:parameter name="makeLegendFigure" value="true" />
    <cfg:parameter name="nGrouping" value="distinct" />
    <cfg:parameter name="kGrouping" value="distinct" />
  </cfg:configuration>

  <e:module class="description.instances.InstanceInformation" />
```

- Global base configuration: 2 figures per row, figure series should have dedicated sub-figure for legend, when benchmarks are grouped either by n or by k, put those with same values of these features together

Listing: Part 1 from file evaluation.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<e:evaluation
  xmlns:e="http://www.optimizationBenchmarking.org/formats/
      evaluationConfiguration/evaluationConfiguration.1.0.xsd"
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/configuration/
      configuration.1.0.xsd">

  <cfg:configuration>
    <cfg:parameter name="figureSize" value="2 per row" />
    <cfg:parameter name="makeLegendFigure" value="true" />
    <cfg:parameter name="nGrouping" value="distinct" />
    <cfg:parameter name="kGrouping" value="distinct" />
  </cfg:configuration>

  <e:module class="description.instances.InstanceInformation" />
```

- Execute one module: print pie charts showing how many benchmark instances have which feature values

Listing: Part 1 from file evaluation.xml for our MAX-SAT example.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<e:evaluation
  xmlns:e="http://www.optimizationBenchmarking.org/formats/
      evaluationConfiguration/evaluationConfiguration.1.0.xsd"
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/configuration/
      configuration.1.0.xsd">

  <cfg:configuration>
    <cfg:parameter name="figureSize" value="2 per row" />
    <cfg:parameter name="makeLegendFigure" value="true" />
    <cfg:parameter name="nGrouping" value="distinct" />
    <cfg:parameter name="kGrouping" value="distinct" />
  </cfg:configuration>

  <e:module class="description.instances.InstanceInformation" />
```

- The ECDF module is applied two times

Listing: Part 2 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>

<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0.01" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
</e:module>
```

- The ECDF module is applied two times: in order to aggregate the ECDF over all problem instances, F is scaled by k and the ECDF is computed for a goal value of $\frac{F}{k} = 0$. The x-axis in *FEs* is log-scaled and figures are rendered page-wide

Listing: Part 2 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>

<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0.01" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
</e:module>
```

- The ECDF module is applied two times: then one ECDF diagram is drawn for each distinct value of n, the log-scaled time measure *RT*, and a goal 0.01 for $\frac{F}{k}$, i.e., for reaching no more than 1% of unsatisfied clauses (and the globally configured figure size)

Listing: Part 2 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>

<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="goal" value="0.01" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
</e:module>
```

- The "Aggregation" module is applied twice as well

Listing: Part 3 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/n)" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="k" />
  </cfg:configuration>
</e:module>

<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="aggregate" value="stddev" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
```

- The “Aggregation” module is applied twice as well: once we plot the median F over runtime measured in *FEs* and divided by n (log-scaled) aggregated over benchmark instances with the same k feature

Listing: Part 3 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/n)" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="k" />
  </cfg:configuration>
</e:module>

<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="aggregate" value="stddev" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
```

- The "Aggregation" module is applied twice as well: then the "standard deviation" is computed, for $\frac{F}{k}$ but this time over the absolute CPU time RT (log-scaled), with one diagram for each distinct value of n

Listing: Part 3 from file evaluation.xml for our MAX-SAT example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/n)" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="k" />
  </cfg:configuration>
</e:module>

<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg RT" />
    <cfg:parameter name="yAxis" value="F/k" />
    <cfg:parameter name="aggregate" value="stddev" />
    <cfg:parameter name="groupBy" value="n" />
  </cfg:configuration>
```

- We now have all the information ready to start an evaluation process

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions

Gluing everything together

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it

- We now have all the information ready to start an evaluation process
    - we specified the measure dimensions
    - we specified the features of the benchmark instances
    - we specified the parameters of our experiments
    - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it
    - Where all of this is

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it
  - Where all of this is
  - What format to use for the report document (LaTeX/PDF? XHTML? Export?)

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it
  - Where all of this is
  - What format to use for the report document (LaTeX/PDF? XHTML? Export?)
  - What kind of figures to generate in the report (PDF? EPS? . . .)

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it
  - Where all of this is
  - What format to use for the report document (LaTeX/PDF? XHTML? Export?)
  - What kind of figures to generate in the report (PDF? EPS? . . .)
  - In case of LaTeX, what document class to use (IEEEtran? sig-alternate? . . .)

- We now have all the information ready to start an evaluation process
  - we specified the measure dimensions
  - we specified the features of the benchmark instances
  - we specified the parameters of our experiments
  - we specified how we want to evaluate the data, what information we want to get
- In order to run the program, we need to tell it
  - Where all of this is
  - What format to use for the report document (LaTeX/PDF? XHTML? Export?)
  - What kind of figures to generate in the report (PDF? EPS? . . .)
  - In case of LaTeX, what document class to use (IEEEtran? sig-alternate? . . .)
- So let's glue everything together

- Use csv+edi as input format (as in our example)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Use csv+edi as input format (as in our example, but we could also use tspSuite or bbob as input format)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify path to input folder, relative to current path

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

Gluing everything together: config.xml

University of Science and Technology of China

- Specify path to input folder, relative to current path (but we could also specify a URL or the path to a ZIP file)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
     configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify path to input folder, relative to current path (but we could also specify a URL or the path to a ZIP file, actually, we can specify multiple paths, URLs, and ZIP files)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Choose LaTeX as output format

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Choose LaTeX as output format (but we could also choose XHTML or export)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Choose LaTeX as output format (but we could also choose XHTML or export, LaTeX documents will automatically be compiled to PDF if LaTeX installation is auto-detected)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Choose PDF as graphics format

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Choose PDF as graphics format (but we could also choose EPS, PNG, TEX, ...)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify output path relative to current directory

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify base name of output document

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- If LaTeX is the output format, specify document class (here IEEEtran)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
     configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

Gluing everything together: `config.xml`

中国科学技术大学 University of Science and Technology of China

- If LaTeX is the output format, specify document class (here `IEEEtran`, but we could also choose `LNCS`, `sig-alternate`, ...)

Listing: Example file `configForIEEEtran.xml` for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify path to evaluation.xml, relative to current directory

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Specify path to evaluation.xml, relative to current directory (but we could also specify a URL or the path to a ZIP file)

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Optional: Tell the system to produce lots of log output to the console and detailed error messages, if any

Listing: Example file configForIEEEtran.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/IEEEtran/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="IEEEtran" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Now let's use the LaTeX document class for Springer's LNCS instead...

Listing: Example file configForLNCS.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="LaTeX" />
  <cfg:parameter name="graphicDriver" value="pdf" />

  <cfg:parameter name="output" value="../reports/LaTeX/LNCS/" />
  <cfg:parameter name="docName" value="report" />
  <cfg:parameter name="documentClass" value="LNCS" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Now let's create an XHTML web page with PNG figures instead...

Listing: Example file configForXHTML.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
     configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="XHTML" />
  <cfg:parameter name="graphicDriver" value="png" />

  <cfg:parameter name="output" value="../reports/XHTML/" />
  <cfg:parameter name="docName" value="report" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

- Now let's export all figures to CSV text files instead, so that we can load them into GnuPlot, MatLab, or whatever for post-processing

Listing: Example file configForExport.xml for our MAX-SAT example.

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg:configuration
  xmlns:cfg="http://www.optimizationBenchmarking.org/formats/
      configuration/configuration.1.0.xsd">

  <cfg:parameter name="inputDriver" value="csv+edi" />
  <cfg:parameter name="inputSource" value="path(../results/)" />

  <cfg:parameter name="documentDriver" value="export" />

  <cfg:parameter name="output" value="../reports/export/" />
  <cfg:parameter name="docName" value="report" />

  <cfg:parameter name="evaluationSetup" value="path(evaluation.xml)" />

  <cfg:parameter name="logger" value="global;ALL" />
</cfg:configuration>
```

1. Now we can finally execute the `optimizationBenchmarking` Evaluator

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml`

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml`

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForXHTML.xml`

Execute optimizationBenchmarking

1. Now we can finally execute the optimizationBenchmarking Evaluator
2. Open a new terminal (command line)
3. cd into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForXHTML.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForExport.xml`

Execute optimizationBenchmarking

1. Now we can finally execute the optimizationBenchmarking Evaluator
2. Open a new terminal (command line)
3. cd into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForXHTML.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForExport.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=whatever.xml`

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForXHTML.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForExport.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=whatever.xml`
5. . . . and that's it.

1. Now we can finally execute the `optimizationBenchmarking` Evaluator
2. Open a new terminal (command line)
3. `cd` into the directory with the configuration file
4. Then execute:
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForIEEEtran.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForLNCS.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForXHTML.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=configForExport.xml` or
   - `java -jar optimizationBenchmarking-0.8.4-full.jar -configXML=whatever.xml`
5. . . . and that's it.
6. Requirement: Java 1.7

- The Evaluator will now produce report documents containing the requested information (and figures)

Result

- The Evaluator will now produce report documents containing the requested information (and figures)

1

Evaluation Report on Six Experiments

Anne Anonymous

***Abstract*—This is the evaluation report on six experiments, namely 1FlipHC, 1FlipHCrs, 2FlipHC, 2FlipHCrs, mFlipHC, and mFlipHCrs on 100 benchmark instances. This report has been generated with the version 0.8.4 of the Evaluator Component of the Optimization Benchmarking Tool Suite.**

I. Instance Information

Experiments were conducted on 100 benchmark instances, which can be distinguished by two features.

The benchmark instances are characterized by two features:

- n (ten values, ranging from 20 to 250)
- k (ten values, ranging from 91 to 1065)

In Figure 2 we illustrate the relative amount of benchmark instances per feature value over all 100 benchmark instances. The slices in the pie charts are the bigger, the more benchmark instances have the associated feature value in comparison to the other values. The more similar the pie sizes are, the more evenly are the benchmark instances distributed over the benchmark feature values, which may be a good idea for fair experimentation.

II. Performance Comparisons

A. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] computed based on $\frac{F}{k}$ over $\log_{10} FEs$. The $\mathrm{ECDF}\left(FEs, \frac{F}{k} \leq 0\right)$ represents the fraction of runs which reach a value of $\frac{F}{k}$ less than or equal to 0 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better.

The corresponding plot is illustrated in Figure 1.

B. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] computed based on $\frac{F}{k}$ over $\log_{10} RT$. The $\mathrm{ECDF}\left(RT, \frac{F}{k} \leq 0.01\right)$ represents the fraction of runs which reach a value of $\frac{F}{k}$ less than or equal to 0.01 for a given ellapsed runtime measured in *RT*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *RT* directly, but instead $\log_{10} RT$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature n grouped together.

The corresponding plots are illustrated in Figure 3.

C. Median of Medians

We analyze the median of medians (*med med*) of F over $\log_{10}\left(\frac{FEs}{n}\right)$. The $\mathrm{med\ med}(FEs, F)$ represents the median of the F for a given ellapsed runtime measured in *FEs*. The median is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate these medians by computing their median. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10}\left(\frac{FEs}{n}\right)$. The instance run sets belonging to instances with the same value of the feature k grouped together.

The corresponding plots are illustrated in Figure 4.

D. Median of Standard Deviations

We analyze the median of standard deviations (*med stddev*) computed based on $\frac{F}{k}$ over $\log_{10} RT$. The $\mathrm{med\ stddev}\left(RT, \frac{F}{k}\right)$ represents the standard deviation of the $\frac{F}{k}$ for a given ellapsed runtime measured in *RT*. The standard deviation is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate these standard deviations by computing their median. The x-axis does not represent the values of *RT* directly, but instead $\log_{10} RT$. The instance run sets belonging to instances with the same value of the feature n grouped together.

The corresponding plots are illustrated in Figure 5.

References

[1] H. H. Hoos and T. Stützle, "Evaluating las vegas algorithms — pitfalls and remedies," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, G. F. Cooper and S. Moral, Eds. Madison, WI, USA: San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 24–26, 1998, pp. 238–245. [Online]. Available: http://www.intellektik.informatik.tu-darmstadt.de/TR/1998/98-02.ps.Z

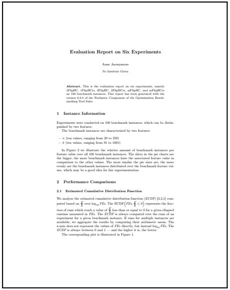
[2] D. A. D. Tompkins and H. H. Hoos, "Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat," in *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, ser. Lecture Notes in Computer Science (LNCS), H. H. Hoos and D. G. Mitchell, Eds., vol. 3542. Vancouver, BC, Canada: Berlin, Germany: Springer-Verlag GmbH, May 10–13, 2004, pp. 306–320. [Online]. Available: http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf?attredirects=0

[3] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep., Mar. 24, 2012. [Online]. Available: http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocexperiment.pdf

first page of the report in LaTeX for `IEEEtran`

- The Evaluator will now produce report documents containing the requested information (and figures)

first page of the report in LaTeX for `IEEEtran`



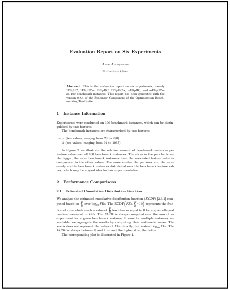
first page of the report in LaTeX for `LNCS`

- The Evaluator will now produce report documents containing the requested information (and figures)


Evaluation Report on Six Experiments

Anne Anonymous

first page of the report in LaTeX for `IEEEtran`


Evaluation Report on Six Experiments

Anne Anonymous

No Institute Given

first page of the report in LaTeX for `LNCS`


Evaluation Report on Six Experiments

Anne Anonymous

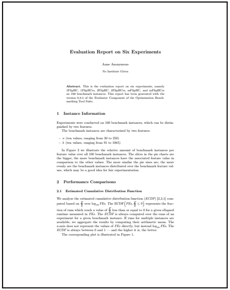
first page of the report in LaTeX for `sig-alternate`

Result

- The Evaluator will now produce report documents containing the requested information (and figures)


Evaluation Report on Six Experiments

Anne Anonymous

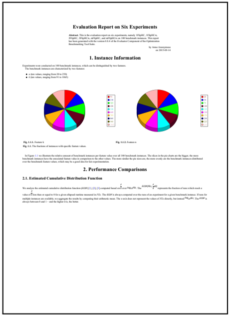

Evaluation Report on Six Experiments

Anne Anonymous

No Institute Given


Evaluation Report on Six Experiments

Anne Anonymous


Evaluation Report on Six Experiments

first page of the report in LaTeX for `IEEEtran`	first page of the report in LaTeX for `LNCS`	first page of the report in LaTeX for `sig-alternate`	first page of the report in `XHTML`

1. Implement your optimization or Machine Learning or whatever algorithm

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances

Usage Summary

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files

Usage Summary

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files
4. Put `dimensions.xml` into results folder

Usage Summary

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files
4. Put `dimensions.xml` into results folder
5. Put `instances.xml` into results folder

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files
4. Put `dimensions.xml` into results folder
5. Put `instances.xml` into results folder
6. Put one `experiment.xml` into each experiment output folder

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files
4. Put `dimensions.xml` into results folder
5. Put `instances.xml` into results folder
6. Put one `experiment.xml` into each experiment output folder
7. Define your evaluation process in a file `evaluation.xml`

1. Implement your optimization or Machine Learning or whatever algorithm
2. Select a well-known set of benchmark instances
3. Run experiments and obtain one output folder per experiment with log files
4. Put `dimensions.xml` into results folder
5. Put `instances.xml` into results folder
6. Put one `experiment.xml` into each experiment output folder
7. Define your evaluation process in a file `evaluation.xml`
8. Execute `optimizationBenchmarking` evaluator

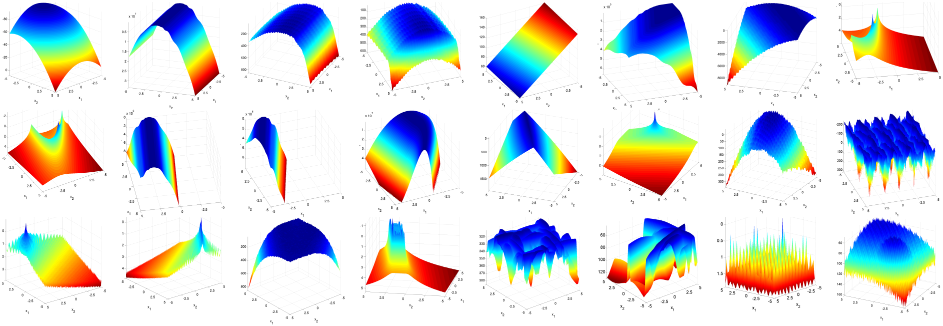
1. Introduction
2. Example 1: MAX-SAT
3. Example 2: BBOB
4. Conclusions

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)

中国科学技术大学 University of Science and Technology of China

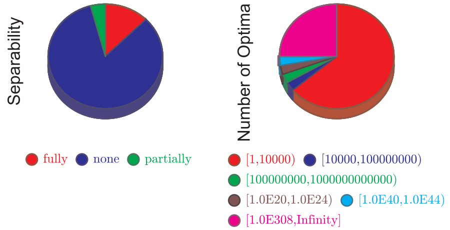
- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems



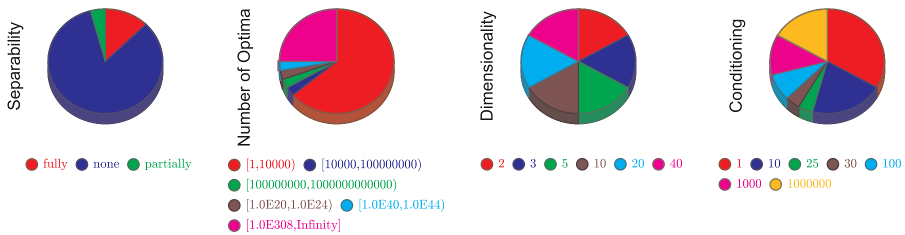
(figures taken from [82])

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension



The relative amounts of *BBOB* benchmark functions according to their features.

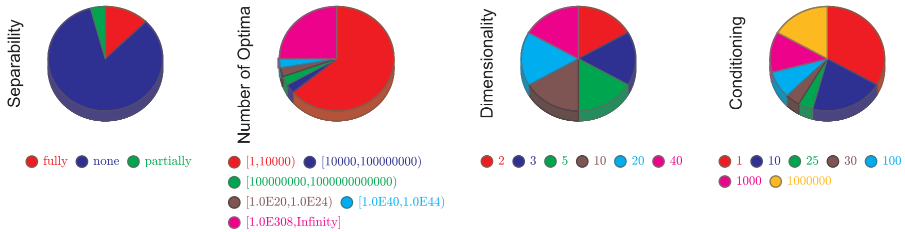
- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability



The relative amounts of *BBOB* benchmark functions according to their features.

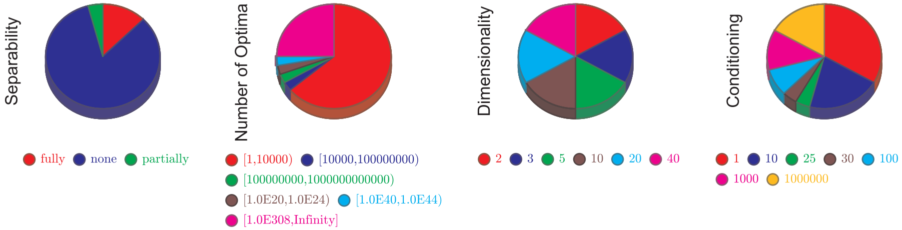
BBOB

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning



The relative amounts of *BBOB* benchmark functions according to their features.

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning, etc.



The relative amounts of *BBOB* benchmark functions according to their features.

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning, etc.
- *COCO* can automatically run experiments, collect log files, and evaluate them

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning, etc.
- *COCO* can automatically run experiments, collect log files, and evaluate them
- The framework and the results of past *BBOB*s are available at `http://coco.gforge.inria.fr`

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning, etc.
- *COCO* can automatically run experiments, collect log files, and evaluate them
- The framework and the results of past *BBOB*s are available at `http://coco.gforge.inria.fr`
- `optimizationBenchmarking` has an experimental input driver for *COCO* data

- Since 2009, the *Black-Box Optimization Benchmarking* (*BBOB*) workshops [71, 80–82] regularly take place at GECCO (now also at CEC)
- Researchers can use the *COmparing Continuous Optimisers* (*COCO*) framework to benchmark their *numerical* optimization algorithms
- *COCO*/*BBOB* defines a set of 24 numerical optimization problems, which differ in features such as dimension, degree of separability, conditioning, etc.
- *COCO* can automatically run experiments, collect log files, and evaluate them
- The framework and the results of past *BBOB*s are available at `http://coco.gforge.inria.fr`
- `optimizationBenchmarking` has an experimental input driver for *COCO* data
- No need to specify `dimensions.xml` and `instances.xml`, as these are fixed and known for *COCO*/*BBOB*.

- You can quickly download all example data and the Evaluator and run the example on your PC by executing the following code snippet.

- You can quickly download all example data and the Evaluator and run the example on your PC by executing the following code snippet.
- System Requirements:
  - Linux (for `make.sh`), Windows (for `make.bat`, tested: Win 8, should work also under Win 7)
  - Java 1.7 (ideally a `JDK` under a `JRE` slower and higher memory consumption)
  - `svn`
  - optional: a LaTeX installation, such as TeXLive (needed for generating pdf reports)

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

Listing: Linux: script `make.sh` for downloading & running the *BBOB* example.

```
#!/bin/bash

jarName="optimizationBenchmarking-full.jar"
bbobDownloadBaseURL="http://coco.lri.fr/BBOB2013/rawdata"

outputDir=`pwd`
echo "Writing output to folder '${outputDir}'"

echo "Downloading selected experimental results from '${bbobDownloadBaseURL}'."
mkdir -p "${outputDir}/results"
cd "${outputDir}/results"
for archive in "hutter2013_CMAES.tar.gz" "liao2013_IPOP.tar.gz" "liao2013_IPOP-500.tar.gz" "liao2013_IPOP-tany.tar.gz" \
               "liao2013_IPOP-texp.tar.gz" "tran2013_P-DCN.tar.gz" "pal2013_DE.tar.gz" "pal2013_fmincon.tar.gz" \
               "pal2013_simplex.tar.gz" "pal2013_HMLSL.tar.gz" "holtschulte2013_hill.tar.gz" "holtschulte2013_ga100.tar.gz"
do
  wget -O "${outputDir}/results/${archive}" "${bbobDownloadBaseURL}/$archive"
  tar -xvf "${outputDir}/results/${archive}"
  rm "${outputDir}/results/${archive}"
done

echo "Downloading evaluation/configuration via 'svn export' from GitHub."
cd "${outputDir}"
svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/bbob/evaluation

jarDownloadURL=$(wget "http://optimizationbenchmarking.github.io/optimizationBenchmarking/currentVersion.url" -q -O -)
echo "Downloading evaluator from '${jarDownloadURL}'."
wget -O "${outputDir}/${jarName}" "${jarDownloadURL}"

echo "Applying evaluator and obtaining report in IEEEtran format."
cd "${outputDir}/evaluation"
java -jar "${outputDir}/${jarName}" -configXML=configForIEEEtran.xml

cd "${outputDir}"
echo "Done."
```

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)

Listing: Windows: script `make.sh` for downloading & running the *BBOB* example.

```
echo "Downloading evaluator."
powershell -command "& {iwr http://optimizationbenchmarking.github.io/optimizationBenchmarking/currentVersion.url -OutFile version.txt}"
for /F "delims=" %i in (version.txt) do set downloadURL=%i
powershell -command "& {iwr %downloadURL% -OutFile optimizationBenchmarking.jar}"
del version.txt

echo "Downloading (but not installing!) required 3rd-party software: downloading SVN client and 7-Zip to extract it."
md svn
cd svn
powershell -command "& {iwr https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/raw/master/tools/windows/7zip/7za.exe -OutFile 7za.exe}"
powershell -command "& {iwr https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/raw/master/tools/windows/svn/svn.tar.lzma -OutFile svn.tar.lzma}"
7za x svn.tar.lzma
7za x svn.tar
cd..

echo "Downloading experimental results from http://coco.lri.fr/BBOB2013/rawdata/
md results
cd results
for %i in (hutter2013_CMAES.tar liao2013_IPOP.tar liao2013_IPOP-500.tar liao2013_IPOP-tany.tar ^
liao2013_IPOP-texp.tar tran2013_P-DCN.tar pal2013_DE.tar pal2013_fmincon.tar ^
pal2013_simplex.tar pal2013_HMLSL.tar holtschulte2013_hill.tar holtschulte2013_ga100.tar) do ^
powershell -command "& { iwr http://coco.lri.fr/BBOB2013/rawdata/%i.gz -OutFile %i.gz }" && ^
..\svn\7za x %i.gz && ^
..\svn\7za x %i && ^
del %i.gz && ^
del %i

cd ..

echo "Downloading evaluation/configuration via 'svn export' from GitHub."
svn\svn export https://github.com/optimizationBenchmarking/optimizationBenchmarkingDocu/branches/master/examples/bbob/evaluation

rd /s /q svn

echo "Applying evaluator and obtaining report in IEEEtran format."
cd evaluation
java -jar "..\optimizationBenchmarking.jar" -configXML=configForIEEEtran.xml

cd..
echo "Done."
```

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)
- After the script, you will have
  - a folder `results` with the log files which have been evaluated
  - a folder `evaluation` with the configuration files and the `evaluation.xml` file defining what to do
  - a filder `reports` with the generated reports

Quick Guide

- Enter (or create) a folder where you want to have everything, then execute this script via copy-paste to the terminal (it may need quite a while to run due to the downloads)
- After the script, you will have
  - a folder `results` with the log files which have been evaluated
  - a folder `evaluation` with the configuration files and the `evaluation.xml` file defining what to do
  - a filder `reports` with the generated reports
- But now, let's continue with the example...

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]
  6. Multi-Objectivization with NSGA-II [102] `tran2013_P-DCN.tar.gz` [103]

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]
  6. Multi-Objectivization with NSGA-II [102] `tran2013_P-DCN.tar.gz` [103]
  7. Differential Evolution (DE): `pal2013_DE.tar.gz` [104]
  8. Quasi-Newton Type Algorithm: `pal2013_fmincon.tar.gz` [105]
  9. Nelder-Mead Simplex [106]: `pal2013_simplex.tar.gz` [105]
  10. Hybrid Multi-Level Single Linkage Algorithm (HMLSL): `pal2013_HMLSL.tar.gz` [104]

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]
  6. Multi-Objectivization with NSGA-II [102]`tran2013_P-DCN.tar.gz` [103]
  7. Differential Evolution (DE): `pal2013_DE.tar.gz` [104]
  8. Quasi-Newton Type Algorithm: `pal2013_fmincon.tar.gz` [105]
  9. Nelder-Mead Simplex [106]: `pal2013_simplex.tar.gz` [105]
  10. Hybrid Multi-Level Single Linkage Algorithm (HMLSL): `pal2013_HMLSL.tar.gz` [104]
  11. Hill Climber: `holtschulte2013_hill.tar.gz` [107]
  12. Generational GA: `holtschulte2013_ga100.tar.gz` [107]

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]
  6. Multi-Objectivization with NSGA-II [102]`tran2013_P-DCN.tar.gz` [103]
  7. Differential Evolution (DE): `pal2013_DE.tar.gz` [104]
  8. Quasi-Newton Type Algorithm: `pal2013_fmincon.tar.gz` [105]
  9. Nelder-Mead Simplex [106]: `pal2013_simplex.tar.gz` [105]
  10. Hybrid Multi-Level Single Linkage Algorithm (HMLSL): `pal2013_HMLSL.tar.gz` [104]
  11. Hill Climber: `holtschulte2013_hill.tar.gz` [107]
  12. Generational GA: `holtschulte2013_ga100.tar.gz` [107]
- We can directly download them from `http://coco.lri.fr/BBOB2013/rawdata`

- We select a set of experiments from the *BBOB* 2013 workshop for evaluation with the `optimizationBenchmarking` Evaluator:
  1. CMA-ES: `hutter2013_CMAES.tar.gz` [99]
  2. IPOP-CMA-ES: `liao2013_IPOP.tar.gz` [100]
  3. IPOP-CMA-ES: `liao2013_IPOP-500.tar.gz` [100]
  4. IPOP-CMA-ES: `liao2013_IPOP-tany.tar.gz` [101]
  5. IPOP-CMA-ES: `liao2013_IPOP-texp.tar.gz` [101]
  6. Multi-Objectivization with NSGA-II [102]`tran2013_P-DCN.tar.gz` [103]
  7. Differential Evolution (DE): `pal2013_DE.tar.gz` [104]
  8. Quasi-Newton Type Algorithm: `pal2013_fmincon.tar.gz` [105]
  9. Nelder-Mead Simplex [106]: `pal2013_simplex.tar.gz` [105]
  10. Hybrid Multi-Level Single Linkage Algorithm (HMLSL): `pal2013_HMLSL.tar.gz` [104]
  11. Hill Climber: `holtschulte2013_hill.tar.gz` [107]
  12. Generational GA: `holtschulte2013_ga100.tar.gz` [107]
- We can directly download them from `http://coco.lri.fr/BBOB2013/rawdata`...
- ...and unpack them into one common folder

- All we need to supply to the Evaluator is

- All we need to supply to the Evaluator is
  1. the `evaluation.xml` file specifying what kind of information we want to obtain from the experimental data

- All we need to supply to the Evaluator is
  1. the `evaluation.xml` file specifying what kind of information we want to obtain from the experimental data and
  2. the a configuration file (let's call it `configForIEEEtran.xml`) telling the Evaluator where everything is and what document driver or document class to use (guess which).

- All we need to supply to the Evaluator is
  1. the `evaluation.xml` file specifying what kind of information we want to obtain from the experimental data and
  2. the a configuration file (let's call it `configForIEEEtran.xml`) telling the Evaluator where everything is and what document driver or document class to use (guess which).
- We now look at the interesting parts of the `evaluation.xml` file (the file in general has been discussed in the previous example)

- Let's first plot the ECDF aggregated over all benchmark instances

Listing: Part 1 from file `evaluation.xml` for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>
```

- Let's first plot the ECDF aggregated over all benchmark instances
- We set the goal "error" to $1 \cdot 10^{-8}$

Listing: Part 1 from file `evaluation.xml` for our *BBOB* example.

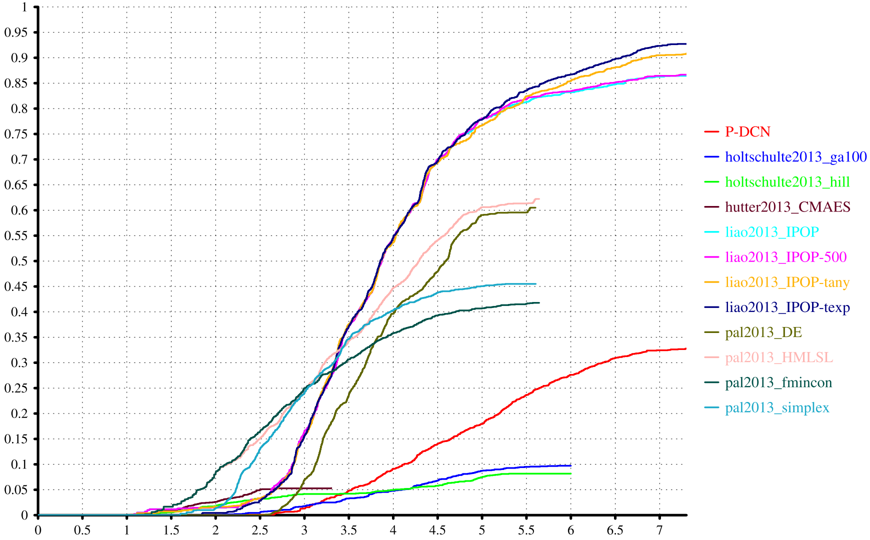
```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>
```

- Let's first plot the ECDF aggregated over all benchmark instances
- We set the goal "error" to $1 \cdot 10^{-8}$
- For the time measured in *FEs* and log-scaled, we plot the fraction of runs achieving this goal

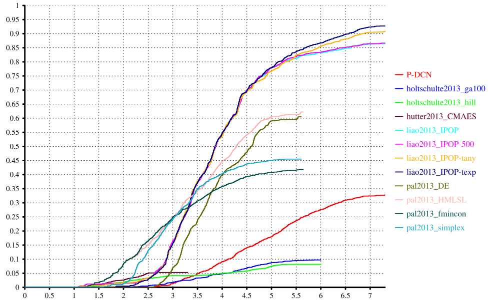
Listing: Part 1 from file evaluation.xml for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="figureSize" value="page wide" />
    <cfg:parameter name="makeLegendFigure" value="false" />
  </cfg:configuration>
</e:module>
```

- Let's first plot the ECDF aggregated over all benchmark instances

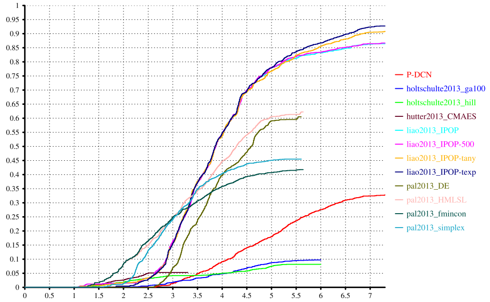


- Let's first plot the ECDF aggregated over all benchmark instances



- It seems that `IPOP-texp` can reach $F \leq 1 \cdot 10^{-8}$ on more instances than the other tested algorithms

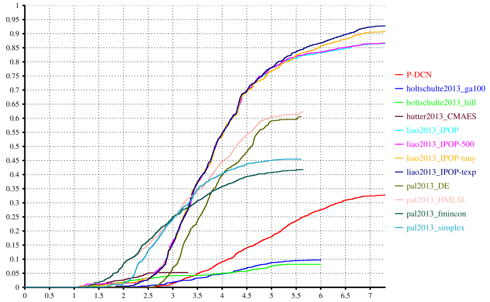
- Let's first plot the ECDF aggregated over all benchmark instances



- It seems that `IPOP-texp` can reach $F \leq 1 \cdot 10^{-8}$ on more instances than the other tested algorithms
- The different `IPOP` variants in general reach this value more often than the other algorithms

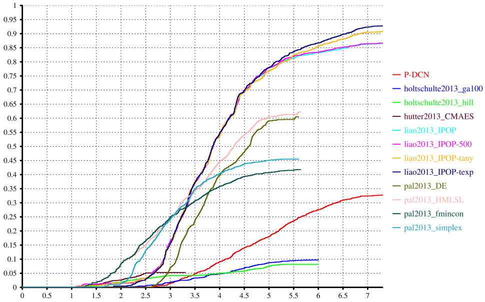
ECDF over Everything

- Let's first plot the ECDF aggregated over all benchmark instances



- It seems that `IPOP-texp` can reach $F \leq 1 \cdot 10^{-8}$ on more instances than the other tested algorithms
- The different `IPOP` variants in general reach this value more often than the other algorithms
- `pal2013_fmincon` and `pal2013_HMLSL` both solve more problems during approximately the first 2500 *FEs*, i.e., are initially faster

- Let's first plot the ECDF aggregated over all benchmark instances



- It seems that `IPOP-texp` can reach $F \leq 1 \cdot 10^{-8}$ on more instances than the other tested algorithms
- The different `IPOP` variants in general reach this value more often than the other algorithms
- `pal2013_fmincon` and `pal2013_HMLSL` both solve more problems during approximately the first 2500 *FEs*, i.e., are initially faster
- The Hill Climber and GA (`holtshulte`) solve the least problems in the comparison

- Let's now plot the ECDF aggregated over each distinct value of the benchmark feature *dimension*

Listing: Part 2 from file `evaluation.xml` for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="groupBy" value="dim" />
  </cfg:configuration>
</e:module>
```

- Let's now plot the ECDF aggregated over each distinct value of the benchmark feature *dimension*
- The goal "error" to achieve is again $1 \cdot 10^{-8}$

Listing: Part 2 from file evaluation.xml for our *BBOB* example.

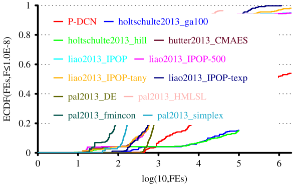
```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="groupBy" value="dim" />
  </cfg:configuration>
</e:module>
```

- Let's now plot the ECDF aggregated over each distinct value of the benchmark feature *dimension*
- The goal "error" to achieve is again $1 \cdot 10^{-8}$ and
- also use the (only) time measured in *FEs*, log-scaled.

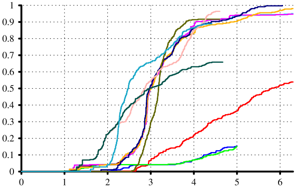
Listing: Part 2 from file `evaluation.xml` for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-8" />
    <cfg:parameter name="groupBy" value="dim" />
  </cfg:configuration>
</e:module>
```

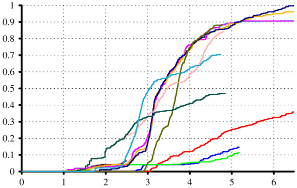
- Let's now plot the ECDF aggregated over each distinct value of the benchmark feature *dimension*



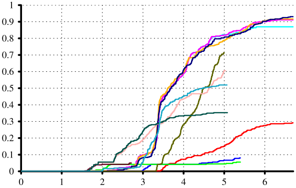
legend



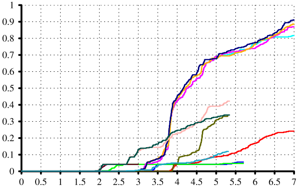
dim = 2



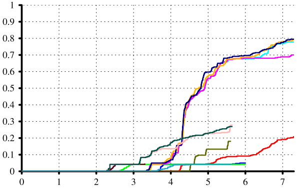
dim = 4



dim = 5



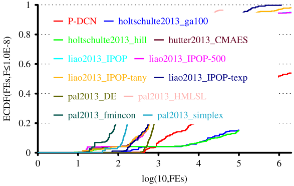
dim = 10



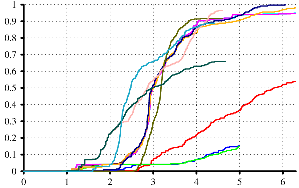
dim = 20

University of Science and Technology of China

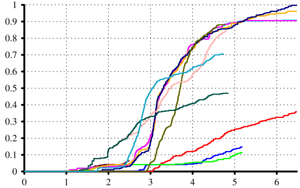
- We find that for larger dimension, fewer problems can be solved



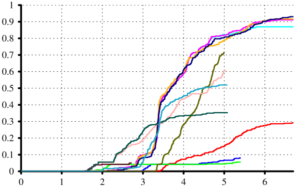
legend



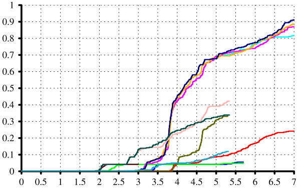
$\mathtt{dim} = 2$



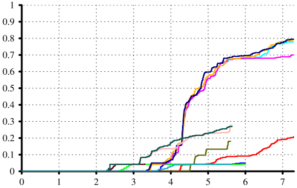
$\mathtt{dim} = 4$



$\mathtt{dim} = 5$

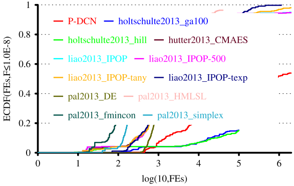


$\mathtt{dim} = 10$

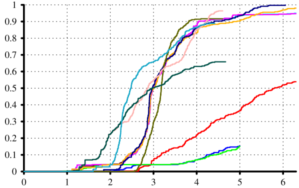


$\mathtt{dim} = 20$

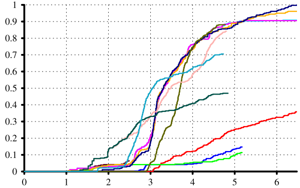
- While the overall performance of `pal2013_fmincon` and `pal2013_simplex` look similar when considering *all* problems, we find that the simplex algorithm is very heavily influenced by the dimension



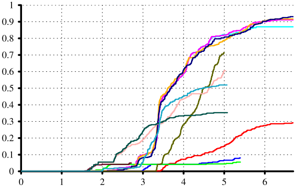
legend



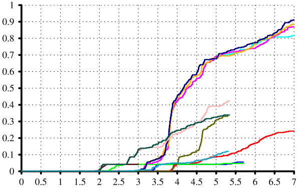
dim = 2



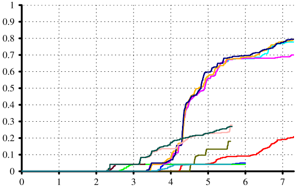
dim = 4



dim = 5

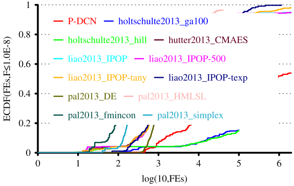


dim = 10

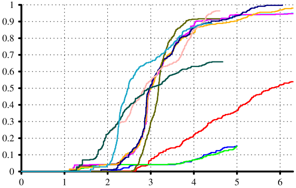


dim = 20

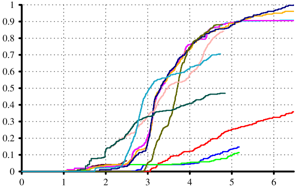
- Similarly, the performance of DE breaks down when the dimension increases



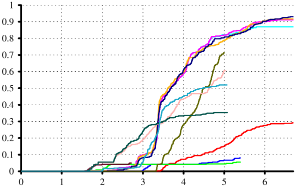
legend



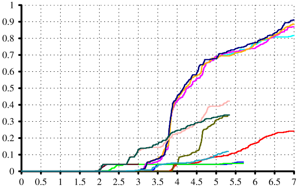
$\mathtt{dim} = 2$



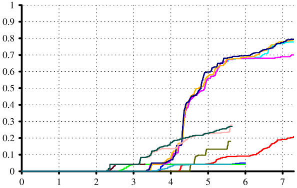
$\mathtt{dim} = 4$



$\mathtt{dim} = 5$



$\mathtt{dim} = 10$

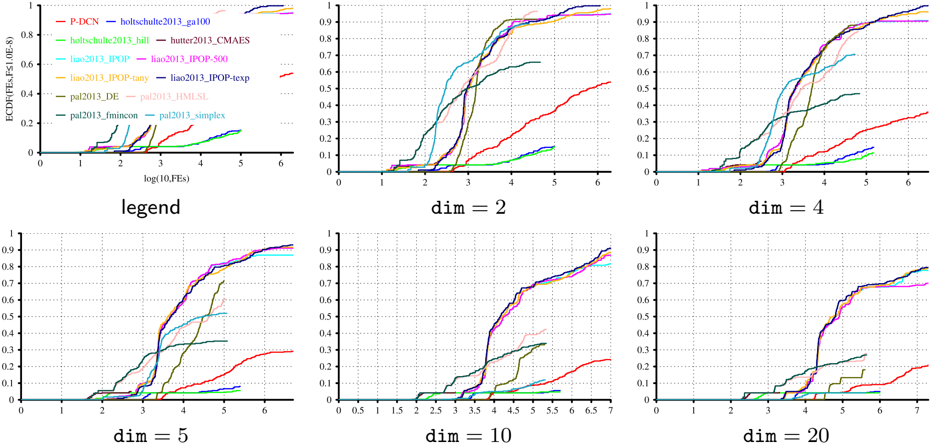


$\mathtt{dim} = 20$

ECDF by Dimension

中国科学技术大学 University of Science and Technology of China

- The performance of the IPOP algorithm family, on the other hand, degenerates gracefully with rising dimension



- Let's now plot the ECDF aggregated over the benchmark instances with the same value of feature *condition number*

Listing: Part 3 from file evaluation.xml for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-5" />
    <cfg:parameter name="groupBy" value="cond" />
  </cfg:configuration>
</e:module>
```

- Let's now plot the ECDF aggregated over the benchmark instances with the same value of feature *condition number*
- *"the condition number corresponds to the square root of the ratio between the largest axis of the ellipsoid and the shortest axis"* [82]

Listing: Part 3 from file evaluation.xml for our *BBOB* example.

```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-5" />
    <cfg:parameter name="groupBy" value="cond" />
  </cfg:configuration>
</e:module>
```

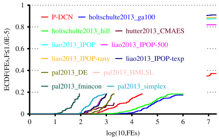
- Let's now plot the ECDF aggregated over the benchmark instances with the same value of feature *condition number*
- *"the condition number corresponds to the square root of the ratio between the largest axis of the ellipsoid and the shortest axis"* [82]
- As goal "error" to achieve, this time we pick $1 \cdot 10^{-5}$

Listing: Part 3 from file `evaluation.xml` for our *BBOB* example.

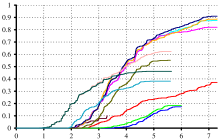
```
<e:module class="all.ecdf.AllECDF">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg FEs" />
    <cfg:parameter name="yAxis" value="F" />
    <cfg:parameter name="goal" value="1e-5" />
    <cfg:parameter name="groupBy" value="cond" />
  </cfg:configuration>
</e:module>
```

ECDF by Condition Number

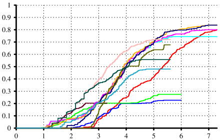
- Let's now plot the ECDF aggregated over the benchmark instances with the same value of feature *condition number*



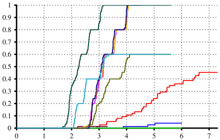
legend



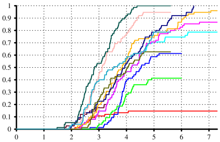
cond $= 1$



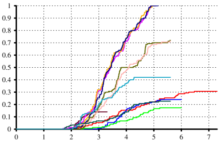
cond $= 10$



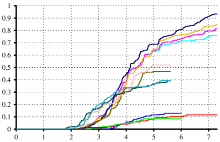
cond $= 25$



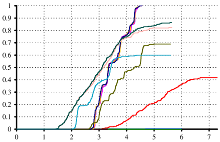
cond $= 30$



cond $= 100$



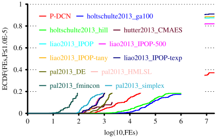
cond $= 1000$



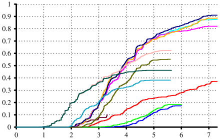
cond $= 1\,000\,000$

ECDF by Condition Number

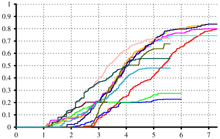
- The influence of the condition number on problem hardness does not seem to obvious at first glance



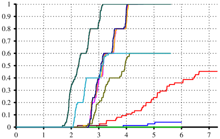
legend



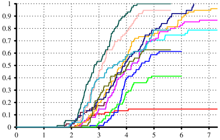
cond $= 1$



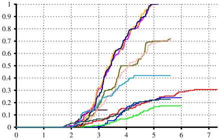
cond $= 10$



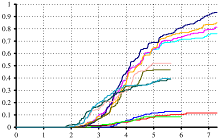
cond $= 25$



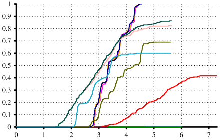
cond $= 30$



cond $= 100$



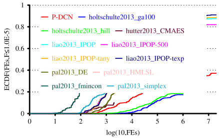
cond $= 1000$



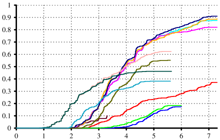
cond $= 1\,000\,000$

ECDF by Condition Number

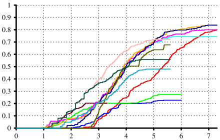
- Some algorithms perform bad on some mediocre condition numbers while performing better on smaller and larger ones (e.g., `P-DCN` on `cond` $= 1000$)



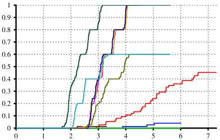
legend



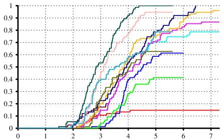
`cond` $= 1$



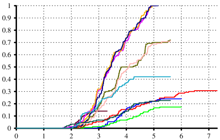
`cond` $= 10$



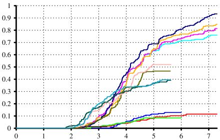
`cond` $= 25$



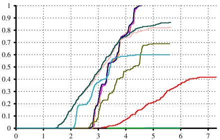
`cond` $= 30$



`cond` $= 100$



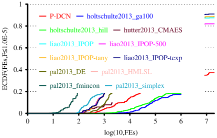
`cond` $= 1000$



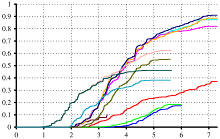
`cond` $= 1\,000\,000$

ECDF by Condition Number

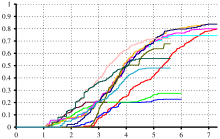
- For some problems, there doesn't seem to be a direct relationship between conditioning and performance (e.g., DE)



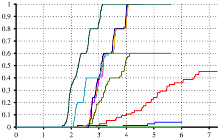
legend



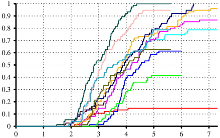
cond = 1



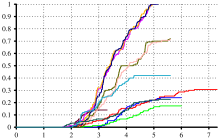
cond = 10



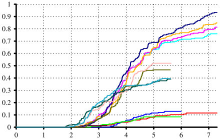
cond = 25



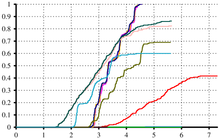
cond = 30



cond = 100

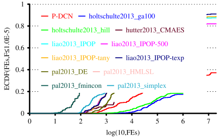


cond = 1000

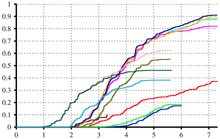


cond = 1 000 000

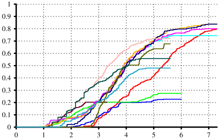
- Possible reason: The problems in the benchmark belonging to a certain condition number may have various other features making them hard or easy



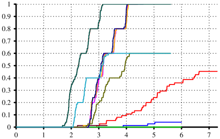
legend



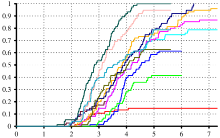
cond = 1



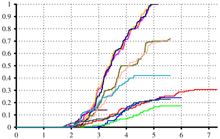
cond = 10



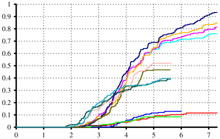
cond = 25



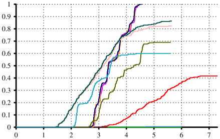
cond = 30



cond = 100

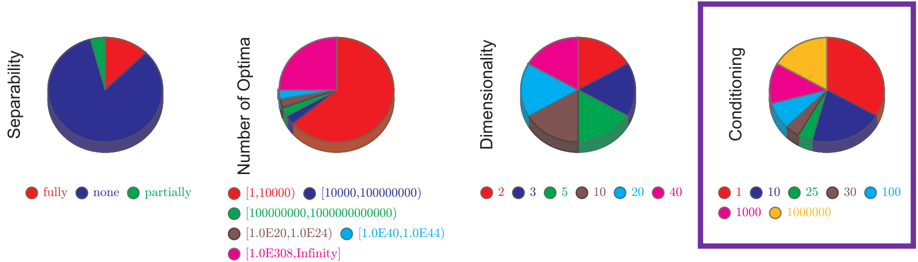


cond = 1000



cond = 1 000 000

- Possible reasons: The problems in the benchmark belonging to a certain condition number may have various other features making them hard or easy and the number of problems per condition number differs largely



The relative amounts of *BBOB* benchmark functions according to their features. (This diagram has also been created with `optimizationBenchmarking`.)

ECDF by Condition Number

- Possible reason: The problems in the benchmark belonging to a certain condition number may have various other features making them hard or easy, the number of problems per condition number differs largely, and the goal value $1 \cdot 10^{-5}$ may be too easy to achieve, leading to a large variance in the results

- Finally, let's see how the algorithms progress on problems of different degrees of separability

Listing: Part 4 from file `evaluation.xml` for our *BBOB* example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/dim²)" />
    <cfg:parameter name="yAxis" value="lg F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="sep" />
  </cfg:configuration>
</e:module>
```

- Finally, let's see how the algorithms progress on problems of different degrees of separability
- The x-axis be again the log-scaled *FEs* divided by the square of the benchmark instance dimension[1]

Listing: Part 4 from file evaluation.xml for our *BBOB* example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/dim^2)" />
    <cfg:parameter name="yAxis" value="lg F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="sep" />
  </cfg:configuration>
</e:module>
```

[1]Yes, the square. Because *why not*. You can do arbitrary mathematical expressions (as long as the preserve the order of the values)

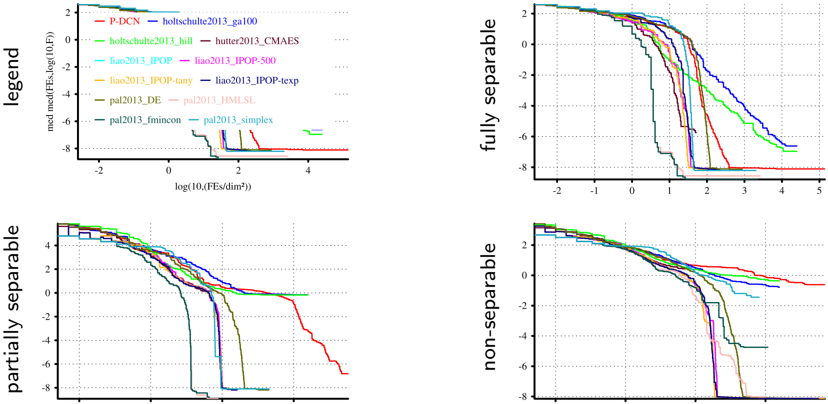
- Finally, let's see how the algorithms progress on problems of different degrees of separability
- The x-axis be again the log-scaled *FEs* divided by the square of the benchmark instance dimension[1] and
- on the y-axis, we plot the median of the log-scaled objective value F

Listing: Part 4 from file `evaluation.xml` for our *BBOB* example.

```
<e:module class="all.aggregation2D.AllAggregation2D">
  <cfg:configuration>
    <cfg:parameter name="xAxis" value="lg(FEs/dim²)" />
    <cfg:parameter name="yAxis" value="lg F" />
    <cfg:parameter name="aggregate" value="median" />
    <cfg:parameter name="groupBy" value="sep" />
  </cfg:configuration>
</e:module>
```

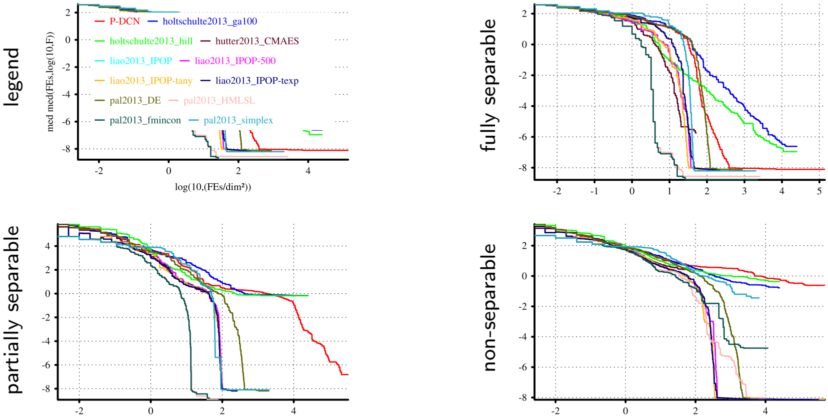
Progress by Separability

- Finally, let's see how the algorithms progress on problems of different degrees of separability

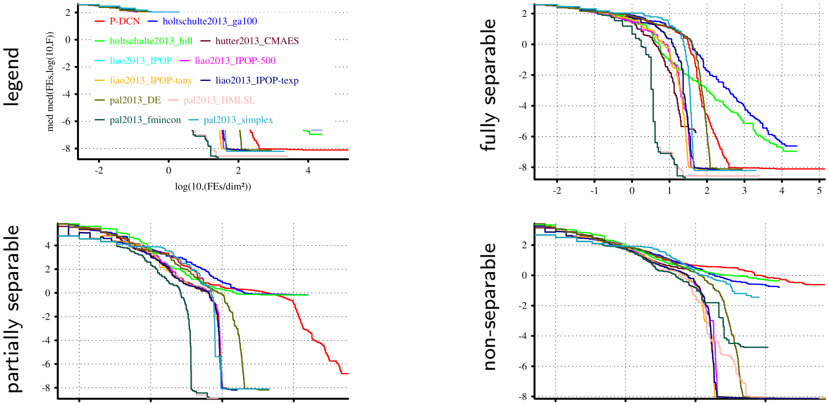


中国科学技术大学
University of Science and Technology of China

- We find that `pal2013_fmincon` and `pal2013_HMLSL` are quite good in solving fully and partially separable problems but both (and especially `pal2013_fmincon`) perform worse on non-separable problems



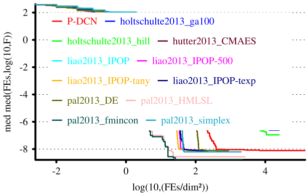
- Here seems to be the strength of the IPOP family of algorithms



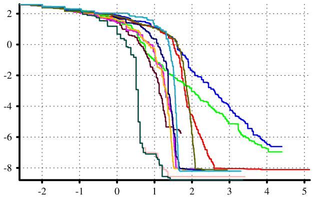
Progress by Separability

- Generally, a decrease in separability, i.e., stronger "variable interactions" [108], makes optimization problems harder for numerical optimization algorithms, which either need longer to or cease to achieve high-quality solutions

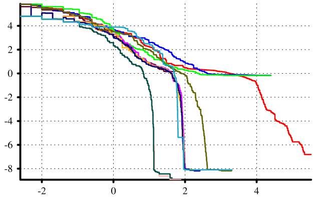
legend



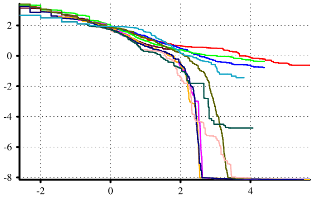
fully separable



partially separable



non-separable



Example Summary

- We can use the `optimizationBenchmarking` Evaluator to analyze data gathered by *COCO* for *BBOB*.

1

Evaluation Report on Twelve Experiments

Anne Anonymous

Abstract—**This is the evaluation report on twelve experiments, namely P-DCN, holtschulte2013_ga100, holtschulte2013_hill, hutter2013_CMAES, liao2013_IPOP, liao2013_IPOP-500, liao2013_IPOP-tany, liao2013_IPOP-texp, pal2013_DE, pal2013_HMLSL, pal2013_fmincon, and pal2013_simplex on 144 benchmark instances. This report has been generated with the version 0.8.4 of the Evaluator Component of the Optimization Benchmarking Tool Suite.**

I. Performance Comparisons

A. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better.

The corresponding plot is illustrated in Figure 1.

B. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *dim* grouped together.

The corresponding plots are illustrated in Figure 2.

C. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}5)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-5 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *cond* grouped together.

The corresponding plots are illustrated in Figure 3.

D. Median of Medians

We analyze the median of medians (*med med*) computed based on $\log_{10} F$ over $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The $\mathrm{med\,med}(FEs, \log_{10} F)$ represents the median of the $\log_{10} F$ for a given ellapsed runtime measured in *FEs*. The median is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate these medians by computing their median. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The instance run sets belonging to instances with the same value of the feature *sep* grouped together.

The corresponding plots are illustrated in Figure 4.

References

[1] H. H. Hoos and T. Stützle, "Evaluating las vegas algorithms — pitfalls and remedies," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, G. F. Cooper and S. Moral, Eds. Madison, WI, USA: San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 24–26, 1998, pp. 238–245. [Online]. Available: http://www.intellektik.informatik.tu-darmstadt.de/TR/1998/98-02.ps.Z

[2] D. A. D. Tompkins and H. H. Hoos, "Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat," in *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, ser. Lecture Notes in Computer Science (LNCS), H. H. Hoos and D. G. Mitchell, Eds., vol. 3542. Vancouver, BC, Canada: Berlin, Germany: Springer-Verlag GmbH, May 10–13, 2004, pp. 306–320. [Online]. Available: http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf?attredirects=0

[3] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep., Mar. 24, 2012. [Online]. Available: http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocexperiment.pdf

first page of the report in LaTeX for `IEEEtran`

- We can use the `optimizationBenchmarking` Evaluator to analyze data gathered by *COCO* for *BBOB*.
- Benchmark instances can be grouped according to features, allowing for convinient analysis of an algorithm's strengths and weaknesses.

1

Evaluation Report on Twelve Experiments

Anne Anonymous

***Abstract*—This is the evaluation report on twelve experiments, namely P-DCN, holtschulte2013_ga100, holtschulte2013_hill, hutter2013_CMAES, liao2013_IPOP, liao2013_IPOP-500, liao2013_IPOP-tany, liao2013_IPOP-texp, pal2013_DE, pal2013_HMLSL, pal2013_fmincon, and pal2013_simplex on 144 benchmark instances. This report has been generated with the version 0.8.4 of the Evaluator Component of the Optimization Benchmarking Tool Suite.**

I. Performance Comparisons

A. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better.

The corresponding plot is illustrated in Figure 1.

B. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *dim* grouped together.

The corresponding plots are illustrated in Figure 2.

C. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}5)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-5 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *cond* grouped together.

The corresponding plots are illustrated in Figure 3.

D. Median of Medians

We analyze the median of medians (*med med*) computed based on $\log_{10} F$ over $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The $\mathrm{med\ med}(FEs, \log_{10} F)$ represents the median of the $\log_{10} F$ for a given ellapsed runtime measured in *FEs*. The median is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate these medians by computing their median. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The instance run sets belonging to instances with the same value of the feature *sep* grouped together.

The corresponding plots are illustrated in Figure 4.

References

[1] H. H. Hoos and T. Stützle, "Evaluating las vegas algorithms — pitfalls and remedies," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, G. F. Cooper and S. Moral, Eds. Madison, WI, USA: San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 24–26, 1998, pp. 238–245. [Online]. Available: http://www.intellektik.informatik.tu-darmstadt.de/TR/1998/98-02.ps.Z

[2] D. A. D. Tompkins and H. H. Hoos, "Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat," in *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, ser. Lecture Notes in Computer Science (LNCS), H. H. Hoos and D. G. Mitchell, Eds., vol. 3542. Vancouver, BC, Canada: Berlin, Germany: Springer-Verlag GmbH, May 10–13, 2004, pp. 306–320. [Online]. Available: http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf?attredirects=0

[3] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep., Mar. 24, 2012. [Online]. Available: http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocexperiment.pdf

first page of the report in LaTeX for `IEEEtran`

- We can use the `optimizationBenchmarking` Evaluator to analyze data gathered by *COCO* for *BBOB*.
- Benchmark instances can be grouped according to features, allowing for convinient analysis of an algorithm's strengths and weaknesses.
- Evaluator modules implemented once can be used for benchmark data from various algorithms and various optimization problems.


1

Evaluation Report on Twelve Experiments

Anne Anonymous

Abstract—**This is the evaluation report on twelve experiments, namely P-DCN, holtschulte2013_ga100, holtschulte2013_hill, hutter2013_CMAES, liao2013_IPOP, liao2013_IPOP-500, liao2013_IPOP-tany, liao2013_IPOP-texp, pal2013_DE, pal2013_HMLSL, pal2013_fmincon, and pal2013_simplex on 144 benchmark instances. This report has been generated with the version 0.8.4 of the Evaluator Component of the Optimization Benchmarking Tool Suite.**

I. Performance Comparisons

A. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better.

The corresponding plot is illustrated in Figure 1.

B. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}8)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-8 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *dim* grouped together.

The corresponding plots are illustrated in Figure 2.

C. Estimated Cumulative Distribution Function

We analyze the estimated cumulative distribution function (*ECDF*) [1], [2], [3] of F over $\log_{10} FEs$. The $\mathrm{ECDF}(FEs, F \leq 1.E\text{-}5)$ represents the fraction of runs which reach a value of F less than or equal to 1.E-5 for a given ellapsed runtime measured in *FEs*. The *ECDF* is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate the results by computing their arithmetic mean. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10} FEs$. The *ECDF* is always between 0 and 1 — and the higher it is, the better. The instance run sets belonging to instances with the same value of the feature *cond* grouped together.

The corresponding plots are illustrated in Figure 3.

D. Median of Medians

We analyze the median of medians (*med med*) computed based on $\log_{10} F$ over $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The $\mathrm{med\,med}(FEs, \log_{10} F)$ represents the median of the $\log_{10} F$ for a given ellapsed runtime measured in *FEs*. The median is always computed over the runs of an experiment for a given benchmark instance. If runs for multiple instances are available, we aggregate these medians by computing their median. The x-axis does not represent the values of *FEs* directly, but instead $\log_{10}\left(\frac{FEs}{dim^2}\right)$. The instance run sets belonging to instances with the same value of the feature *sep* grouped together.

The corresponding plots are illustrated in Figure 4.

References

[1] H. H. Hoos and T. Stützle, "Evaluating las vegas algorithms — pitfalls and remedies," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, G. F. Cooper and S. Moral, Eds. Madison, WI, USA: San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 24–26, 1998, pp. 238–245. [Online]. Available: http://www.intellektik.informatik.tu-darmstadt.de/TR/1998/98-02.ps.Z

[2] D. A. D. Tompkins and H. H. Hoos, "Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat," in *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, ser. Lecture Notes in Computer Science (LNCS), H. H. Hoos and D. G. Mitchell, Eds., vol. 3542. Vancouver, BC, Canada: Berlin, Germany: Springer-Verlag GmbH, May 10–13, 2004, pp. 306–320. [Online]. Available: http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf?attredirects=0

[3] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Tech. Rep., Mar. 24, 2012. [Online]. Available: http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocexperiment.pdf

first page of the report in LaTeX for `IEEEtran`

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*

Conclusions

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*
- But it can already load and evaluate performance data from *your* optimization or Machine Learning algorithm

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*
- But it can already load and evaluate performance data from *your* optimization or Machine Learning algorithm
- It can help *you* to understand what the strengths and weaknesses of *your* algorithm are

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*
- But it can already load and evaluate performance data from *your* optimization or Machine Learning algorithm
- It can help *you* to understand what the strengths and weaknesses of *your* algorithm are
- It produces figures ready for use in *your* publication

Conclusions

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*
- But it can already load and evaluate performance data from *your* optimization or Machine Learning algorithm
- It can help *you* to understand what the strengths and weaknesses of *your* algorithm are
- It produces figures ready for use in *your* publication
- ...and these figures are optimized (size, fonts) for the journal or conference *you* want to submit to.

- I have presented a very first version of the Evaluator component of `optimizationBenchmarking`
- It still lacks several features you are used from *TSP Suite* or *COCO*
- But it can already load and evaluate performance data from *your* optimization or Machine Learning algorithm
- It can help *you* to understand what the strengths and weaknesses of *your* algorithm are
- It produces figures ready for use in *your* publication
- ...and these figures are optimized (size, fonts) for the journal or conference *you* want to submit to.
- Btw, you could even compare general algorithms (like GAs and HC) on entirely different problem types at once (like MAX-SAT and *BBOB*) by making the problem type an instance feature...

- Add the missing text to the different evaluation modules

Future Work: Short-Term

- Add the missing text to the different evaluation modules
- Add more modules, to reach *TSP Suite*'s power, e.g., add automated algorithm ranking

Future Work: Short-Term

- Add the missing text to the different evaluation modules
- Add more modules, to reach *TSP Suite*'s power, e.g., add automated algorithm ranking
- Publicize the use `optimizationBenchmarking` about colleagues

Future Work: Short-Term

- Add the missing text to the different evaluation modules
- Add more modules, to reach *TSP Suite*'s power, e.g., add automated algorithm ranking
- Publicize the use `optimizationBenchmarking` about colleagues
- Improve features based on feedback

Future Work: Short-Term

- Add the missing text to the different evaluation modules
- Add more modules, to reach *TSP Suite*'s power, e.g., add automated algorithm ranking
- Publicize the use `optimizationBenchmarking` about colleagues
- Improve features based on feedback
- Write an overview paper about our system to publish it more widely

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules

Future Work: Long-Term

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules

Future Work: Long-Term

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules
- Idea: We could use clustering to group algorithms by their behavior or problems by their hardness

Future Work: Long-Term

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules
- Idea: We could use clustering to group algorithms by their behavior or problems by their hardness
- Idea: We could use Machine Learning to predict algorithm performance or result quality based on problem features

Future Work: Long-Term

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules
- Idea: We could use clustering to group algorithms by their behavior or problems by their hardness
- Idea: We could use Machine Learning to predict algorithm performance or result quality based on problem features
- Idea: We could use regression or curve fitting to find curves fitting to measured progress or ECDF functions and then use these to compare with or develop new theoretical concepts

- Scout for new interesting ways to evaluate optimization and Machine Learning algorithms and implement them as evaluator modules
- Idea: We could use clustering to group algorithms by their behavior or problems by their hardness
- Idea: We could use Machine Learning to predict algorithm performance or result quality based on problem features
- Idea: We could use regression or curve fitting to find curves fitting to measured progress or ECDF functions and then use these to compare with or develop new theoretical concepts
- Btw: This is Big Data, since we can collect *much* information...

Visit our website

http://www.optimizationBenchmarking.org

or

http://optimizationbenchmarking.github.io/optimizationBenchmarking

for downloading the software (version 0.8.4) and obtaining more information.

System Requirements:

- Java 1.7 (Ideally a JDK, under JRE slower with more memory requirements)
- optional: a LaTeX installation, such as TeXLive or MiKTeX (needed for generating pdf reports)

谢谢！

Thank you.

Thomas Weise
tweise@ustc.edu.cn ·
tweise@gmx.de ·
http://www.it-weise.de

USTC-Birmingham Joint Res. Inst. in Intelligent Computation and Its Applications (UBRI)
University of Science and Technology of China (USTC), Hefei 230027, Anhui, China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

1. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Computational Intelligence Library. New York, NY, USA: Oxford University Press, Inc., Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), and Boca Raton, FL, USA: CRC Press, Inc., January 1, 1997. ISBN 0-7503-0392-1, 0-7503-0895-8, 978-0-7503-0392-7, and 978-0-7503-0895-3. URL http://books.google.de/books?id=n5nuiIZvmpAC.
2. Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz, editors. *Variants of Evolutionary Algorithms for Real-World Applications*. Berlin/Heidelberg: Springer-Verlag, 2011. ISBN 978-3-642-23423-1 and 978-3-642-23424-8. doi: 10.1007/978-3-642-23424-8. URL http://books.google.de/books?id=B2ONePP40MEC.
3. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), January 2000. ISBN 0750306645 and 9780750306645. URL http://books.google.de/books?id=4HMYCq9US78C.
4. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), November 2000. ISBN 0750306653 and 9780750306652.
5. Dumitru (Dan) Dumitrescu, Beatrice Lazzerini, Lakhmi C. Jain, and A. Dumitrescu. *Evolutionary Computation*, volume 18 of *International Series on Computational Intelligence*. Boca Raton, FL, USA: CRC Press, Inc., June 2000. ISBN 0-8493-0588-8 and 978-0-8493-0588-7. URL http://books.google.de/books?id=MSU9ep79JvUC.
6. Ágoston E. Eiben, editor. *Evolutionary Computation*. Theoretical Computer Science. Amsterdam, The Netherlands: IOS Press, 1999. ISBN 4-274-90269-2, 90-5199-471-0, 978-4-274-90269-7, and 978-90-5199-471-1. URL http://books.google.de/books?id=8LVAGQAACAAJ. This is the book edition of the journal Fundamenta Informaticae, Volume 35, Nos. 1-4, 1998.
7. David Wolfe Corne, Marco Dorigo, Fred W. Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli, and Kenneth V. Price, editors. *New Ideas in Optimization*. McGraw-Hill's Advanced Topics In Computer Science Series. Maidenhead, England, UK: McGraw-Hill Ltd., May 1999. ISBN 0-07-709506-5 and 978-0-07-709506-2. URL http://books.google.de/books?id=nC35AAAACAAJ.
8. Ashish Ghosh and Shigeyoshi Tsutsui, editors. *Advances in Evolutionary Computing – Theory and Applications*. Natural Computing Series. New York, NY, USA: Springer New York, November 22, 2002. ISBN 3-540-43330-9 and 978-3-540-43330-9. URL http://books.google.de/books?id=OGMEMC9P3vMC.
9. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. Germany: it-weise.de (self-published), 2009. URL http://www.it-weise.de/projects/book.pdf.

10. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(1):29–41, February 1996. doi: 10.1109/3477.484436. URL ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf.
11. Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Books. Cambridge, MA, USA: MIT Press, July 1, 2004. ISBN 0-262-04219-3 and 978-0-262-04219-2. URL http://books.google.de/books?id=_aefcpY8GiEC.
12. Michael Guntsch and Martin Middendorf. Applying population based aco to dynamic optimization problems. In Marco Dorigo, Gianni A. Di Caro, and Michael Samples, editors, *From Ant Colonies to Artificial Ants – Proceedings of the Third International Workshop on Ant Colony Optimization (ANTS'02)*, volume 2463/2002 of *Lecture Notes in Computer Science (LNCS)*, pages 111–122, Brussels, Belgium, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45724-0_10. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.6580.
13. Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 132(1-4):373–395, November 2004. doi: 10.1023/B:ANOR.0000039526.52305.af.
14. Ingo Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Farnborough, Hampshire, UK: Royal Aircraft Establishment, August 1965. Library Translation 1122.
15. Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Berlin, Germany: Technische Universität Berlin, 1971. URL http://books.google.de/books?id=QcNNGQAACAAJ.
16. Ingo Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Bad Cannstadt, Stuttgart, Baden-Württemberg, Germany: Frommann-Holzboog Verlag, 1994. ISBN 3-7728-1642-8 and 978-3-772-81642-0. URL http://books.google.de/books?id=savAAAACAAJ.
17. Hans-Paul Schwefel. Kybernetische evolution als strategie der exprimentellen forschung in der strömungstechnik. Master's thesis, Berlin, Germany: Technische Universität Berlin, 1965.
18. Hans-Paul Schwefel. Experimentelle optimierung einer zweiphasendüse teil i. Technical Report 35, Berlin, Germany: AEG Research Institute, 1968. Project MHD–Staustrahlrohr 11.034/68.
19. Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Berlin, Germany: Technische Universität Berlin, Institut für Meß- und Regelungstechnik, Institut für Biologie und Anthropologie, 1975.
20. Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution – A Practical Approach to Global Optimization*. Natural Computing Series. Basel, Switzerland: Birkhäuser Verlag, 2005. ISBN 3-540-20950-6, 3-540-31306-0, 978-3-540-20950-8, and 978-3-540-31306-9. URL http://books.google.de/books?id=S67vX-KqVqUC.

21. Vitaliy Feoktistov. *Differential Evolution – In Search of Solutions*, volume 5 of *Springer Optimization and Its Applications*. New York, NY, USA: Springer New York, December 2006. ISBN 0-387-36895-7, 0-387-36896-5, 978-0-387-36895-5, and 978-0-387-36896-2. URL http://books.google.de/books?id=kG7aP_v-SU4C.
22. Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos Artemio Coello Coello. A comparative study of differential evolution variants for global optimization. In Maarten Keijzer and Mike Cattolico, editors, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, pages 485–492, Seattle, WA, USA: Renaissance Seattle Hotel, 2006. New York, NY, USA: ACM Press. doi: 10.1145/1143997.1144086. URL http://delta.cs.cinvestav.mx/~ccoello/conferences/mezura-gecco2006.pdf.gz.
23. Janez Brest, Viljem Žumer, and Mirjam Sepesy Maučec. Control parameters in self-adaptive differential evolution. In Bogdan Filipič and Jurij Šilc, editors, *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications (BIOMA'06)*, Informacijska Družba (Information Society), pages 35–44, Ljubljana, Slovenia: Jožef Stefan International Postgraduate School, 2006. Ljubljana, Slovenia: Jožef Stefan Institute. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.106.8106.
24. Jouni A. Lampinen and Ivan Zelinka. On stagnation of the differential evolution algorithm. In Pavel Osmera, editor, *Proceedings of the 6th International Conference on Soft Computing (MENDEL'00)*, pages 76–83, Brno, Czech Republic: Brno University of Technology, 2000. Brno, Czech Republic: Brno University of Technology, Ústav Automatizace a Informatiky. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7932.
25. Roberto R. F. Mendes and Arvind S. Mohais. Dynde: A differential evolution for dynamic optimization problems. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 3, pages 2808–2815, Edinburgh, Scotland, UK, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1555047. URL http://www3.di.uminho.pt/~rcm/publications/DynDE.pdf.
26. Patricia Besson, Jean-Marc Vesin, Vlad Popovici, and Murat Kunt. Differential evolution applied to a multimodal information theoretic optimization problem. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, Ernesto Jorge Fernandes Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason H. Moore, Juan Romero, George D. Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *Applications of Evolutionary Computing – Proceedings of EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC (EvoWorkshops'06)*, volume 3907/2006 of *Lecture Notes in Computer Science (LNCS)*, pages 505–509, Budapest, Hungary, 2006. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/11732242_46.

University of Science and Technology of China

27. Rainer M. Storn. Differential evolution (de) for continuous function optimization (an algorithm by kenneth price and rainer storn), 2010. URL http://www.icsi.berkeley.edu/~storn/code.html.
28. Nikolaus Hansen, Andreas Ostermeier, and Andreas Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 57–64, Pittsburgh, PA, USA: University of Pittsburgh, 1995. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.9321.
29. Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Keisoku Jidō and Seigyo Gakkai, editors, *Proceedings of IEEE International Conference on Evolutionary Computation (CEC'96)*, pages 312–317, Nagoya, Aichi, Japan: Nagoya University, Symposium & Toyoda Auditorium, 1996. Los Alamitos, CA, USA: IEEE Computer Society Press. URL http://www.lri.fr/~hansen/CMAES.pdf.
30. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. URL http://www.bionik.tu-berlin.de/user/niko/cmaartic.pdf.
31. Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003. doi: 10.1162/106365603321828970. URL http://mitpress.mit.edu/journals/pdf/evco_11_1_1_0.pdf.
32. Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In Xin Yao, Edmund K. Burke, José Antonio Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiño, Ata Kabán, and Hans-Paul Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242/2004 of *Lecture Notes in Computer Science (LNCS)*, pages 282–291, Birmingham, UK, 2008. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-30217-9_29. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.163.
33. Nikolaus Hansen. The cma evolution strategy: A comparing review. In José Antonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation – Advances on Estimation of Distribution Algorithms*, volume 192/2006 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Berlin, Germany: Springer-Verlag GmbH, 2006. URL http://www.lri.fr/~hansen/hansenedacomparing.pdf.

34. Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, pages 1769–1776, Edinburgh, Scotland, UK, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554902. URL http://www.lri.fr/~hansen/cec2005ipopcmaes.pdf.
35. Anne Auger and Nikolaus Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1777–1784, Edinburgh, Scotland, UK, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554903. URL http://www.lri.fr/~hansen/cec2005localcmaes.pdf.
36. Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 1558608729 and 978-1558608726. URL http://books.google.de/books?id=3HAedXnC49IC.
37. Emile H. L. Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Princeton, NJ, USA: Princeton University Press, 1997. ISBN 0585277540, 0691115222, 9780585277547, and 9780691115221. URL http://books.google.de/books?id=NWghN9G7q9MC.
38. Matthijs den Besten, Thomas Stützle, and Marco Dorigo. Design of iterated local search algorithms. In Egbert J. W. Boers, Jens Gottlieb, Pier Luca Lanzi, Robert Elliott Smith, Stefano Cagnoni, Emma Hart, Günther R. Raidl, and Harald Tijink, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM (EvoWorkshops'01)*, volume 2037/2001 of *Lecture Notes in Computer Science (LNCS)*, pages 441–451, Lake Como, Milan, Italy, 2001. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45365-2_46.
39. Peter Salamon, Paolo Sibani, and Richard Frost. *Facts, Conjectures, and Improvements for Simulated Annealing*, volume 7 of *SIAM Monographs on Mathematical Modeling and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM), 2002. ISBN 0898715083 and 9780898715088. URL http://books.google.de/books?id=jhAldlYvClcC.
40. Peter J. M. van Laarhoven and Emile H. L. Aarts, editors. *Simulated Annealing: Theory and Applications*, volume 37 of *Mathematics and its Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987. ISBN 90-277-2513-6, 978-90-277-2513-4, and 978-90-481-8438-5. URL http://books.google.de/books?id=-IgUab6Dp_IC.

Bibliography VI

中国科学技术大学
University of Science and Technology of China

41. Lawrence Davis, editor. *Genetic Algorithms and Simulated Annealing*. Research Notes in Artificial Intelligence. London, UK: Pitman, 1987. ISBN 0273087711, 0934613443, 9780273087717, and 978-0934613446. URL http://books.google.de/books?id=edfSSAAACAAJ.
42. James C. Spall. *Introduction to Stochastic Search and Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, first edition, June 2003. ISBN 0-471-33052-3, 0-471-72213-8, 978-0-471-33052-3, and 978-0-471-72213-7. URL http://books.google.de/books?id=f66OIvvkKnAC.
43. Scott Kirkpatrick, Charles Daniel Gelatt, Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science Magazine*, 220(4598):671–680, May 13, 1983. doi: 10.1126/science.220.4598.671. URL http://fezzik.ucd.ie/msc/cscs/ga/kirkpatrick83optimization.pdf.
44. Vladimír Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985. doi: 10.1007/BF00940812. URL http://mkweb.bcgsc.ca/papers/cerny-travelingsalesman.pdf. Communicated by S. E. Dreyfus. Also: Technical Report, Comenius University, Mlynská Dolina, Bratislava, Czechoslovakia, 1982.
45. Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. Monte carlo techniques in code optimization. *ACM SIGMICRO Newsletter*, 13(4):143–148, December 1982.
46. Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. Monte carlo techniques in code optimization. In *International Symposium on Microarchitecture – Proceedings of the 15th Annual Workshop on Microprogramming (MICRO 15)*, pages 143–146, Palo Alto, CA, USA, 1982. Piscataway, NJ, USA: IEEE (Institute of Electrical and Electronics Engineers).
47. Martin Pincus. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research (Oper. Res.)*, 18(6):1225–1228, November–December 1970.
48. Fred W. Glover. Tabu search – part i. *ORSA Journal on Computing*, 1(3):190–206, 1989. doi: 10.1287/ijoc.1.3.190. URL http://leeds-faculty.colorado.edu/glover/TS%20-%20Part%20I-ORSA.pdf.
49. Fred W. Glover. Tabu search – part ii. *ORSA Journal on Computing*, 2(1):190–206, 1990. doi: 10.1287/ijoc.2.1.4. URL http://leeds-faculty.colorado.edu/glover/TS%20-%20Part%20II-ORSA-aw.pdf.
50. Fred W. Glover and Manuel Laguna. Tabu search. In Colin R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series. Chichester, West Sussex, UK: Blackwell Publishing Ltd, 1993. ISBN 079239965X and 978-0470220795. URL http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf.

51. Dominique de Werra and Alain Hertz. Tabu search techniques: A tutorial and an application to neural networks. *OR Spectrum – Quantitative Approaches in Management*, 11(3):131–141, September 1989. doi: 10.1007/BF01720782. URL http://www.springerlink.de/content/x25k97k0qx237553/fulltext.pdf.
52. Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994. doi: 10.1287/ijoc.6.2.126. URL http://citeseer.ist.psu.edu/141556.html.
53. Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program C3P 826, Pasadena, CA, USA: California Institute of Technology (Caltech), Caltech Concurrent Computation Program (C3P), 1989. URL http://www.each.usp.br/sarajane/SubPaginas/arquivos_aulas_IA/memetic.pdf.
54. Pablo Moscato. Memetic algorithms. In Panos M. Pardalos and Mauricio G.C. Resende, editors, *Handbook of Applied Optimization*, chapter 3.6.4, pages 157–167. New York, NY, USA: Oxford University Press, Inc., 2002.
55. Pablo Moscato and Carlos Cotta. A gentle introduction to memetic algorithms. In Fred W. Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 5, pages 105–144. Norwell, MA, USA: Kluwer Academic Publishers, Dordrecht, Netherlands: Springer Netherlands, and Boston, MA, USA: Springer US, 2003. doi: 10.1007/0-306-48056-5_5. URL http://www.lcc.uma.es/~ccottap/papers/handbook03memetic.pdf.
56. Ágoston E. Eiben and James E. Smith. Hybridisation with other techniques: Memetic algorithms. In *Introduction to Evolutionary Computing*, Natural Computing Series, chapter 10, pages 173–188. New York, NY, USA: Springer New York, 2003.
57. William Eugene Hart, Natalio Krasnogor, and James E. Smith, editors. *Recent Advances in Memetic Algorithms*, volume 166/2005 of *Studies in Fuzziness and Soft Computing*. Berlin, Germany: Springer-Verlag GmbH, 2005. ISBN 3-540-22904-3 and 978-3-540-22904-9. doi: 10.1007/3-540-32363-5. URL http://books.google.de/books?id=LYf7YW4DmkUC.
58. Jason Digalakis and Konstantinos Margaritis. Performance comparison of memetic algorithms. *Journal of Applied Mathematics and Computation*, 158:237–252, October 2004. doi: 10.1016/j.amc.2003.08.115. URL http://www.complexity.org.au/ci/draft/draft/digala02/digala02s.pdf.

59. Nicholas J. Radcliffe and Patrick David Surry. Formal memetic algorithms. In Terence Claus Fogarty, editor, *Proceedings of the Workshop on Artificial Intelligence and Simulation of Behaviour, International Workshop on Evolutionary Computing, Selected Papers (AISB'94)*, volume 865/1994 of *Lecture Notes in Computer Science (LNCS)*, pages 1–16, Leeds, UK, 1994. Chichester, West Sussex, UK: Society for the Study of Artificial Intelligence and the Simulation of Behaviour (SSAISB), Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-58483-8_1. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.9885.
60. David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, February 2007. ISBN 0-691-12993-2 and 978-0-691-12993-8. URL http://books.google.de/books?id=nmF4rVNJMVsC.
61. Eugene Leighton (Gene) Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, September 1985. ISBN 0-471-90413-9 and 978-0-471-90413-7. URL http://books.google.de/books?id=BXBGAAAAYAAJ.
62. Gregory Z. Gutin and Abraham P. Punnen, editors. *The Traveling Salesman Problem and its Variations*, volume 12 of *Combinatorial Optimization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN 0-306-48213-4, 1-4020-0664-0, and 978-1-4020-0664-7. doi: 10.1007/b101971. URL http://books.google.de/books?id=TRYkPg_Xf20C.
63. Weiqi Li. Seeking global edges for traveling salesman problem in multi-start search. *Journal of Global Optimization*, 51 (3):515–540, November 2011. doi: 10.1007/s10898-010-9643-4.
64. Sami Khuri, Martin Schütz, and Jörg Heitkötter. Evolutionary heuristics for the bin packing problem. In David W. Pearson, Nigel C. Steele, and Rudolf F. Albrecht, editors, *Proceedings of the 2nd International Conference on Artificial Neural Nets and Genetic Algorithms (ICANNGA'95)*, pages 285–288, Alès, France, 1995. New York, NY, USA: Springer New York. URL http://www6.uniovi.es/pub/EC/GA/papers/icannga95.ps.gz.
65. Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In Ian P. Gent, Hans van Maaren, and Toby Walsh, editors, *SAT2000 – Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 283–292. Amsterdam, The Netherlands: IOS Press, 2000. URL http://www.cs.ubc.ca/~hoos/Publ/sat2000-satlib.pdf.

66. Dave Andrew Douglas Tompkins and Holger H. Hoos. Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat. In Holger H. Hoos and David G. Mitchell, editors, *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3542 of *Lecture Notes in Computer Science (LNCS)*, pages 306–320, Vancouver, BC, Canada, 2004. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/11527695_24. URL http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf.
67. Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter Burkhard, Walter Savitch, Emily P. Friedman, and Alfred Vaino Aho, editors, *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226, San Diego, CA, USA, 1978. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/800133.804350. URL http://www.ccs.neu.edu/home/lieber/courses/csg260/f06/materials/papers/max-sat/p216-schaefer.pdf.
68. Claudio Rossi, Elena Marchiori, and Joost N. Kok. An adaptive evolutionary algorithm for the satisfiability problem. In *Proceedings of the 2000 ACM symposium on Applied computing (SAC'00)*, volume 1, pages 463–469, Villa Olmo, Como, Italy, 2000. New York, NY, USA: ACM Press. doi: 10.1145/335603.335912. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.4771.
69. Peter Merz and Bernd Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In Peter John Angeline, Zbigniew Michalewicz, Marc Schoenauer, Xin Yao, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 2063–2070, Washington, DC, USA: Mayflower Hotel, 1999. Piscataway, NJ, USA: IEEE Computer Society. URL http://en.scientificcommons.org/204950.
70. Luca Maria Gambardella, Éric D. Taillard, and Marco Dorigo. Ant colonies for the quadratic assignment problem. *The Journal of the Operational Research Society (JORS)*, 50(2):167–176, February 1999. doi: 10.2307/3010565. URL http://www.idsia.ch/~luca/tr-idsia-4-97.pdf.
71. Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Rapports de Recherche 7215, Institut National de Recherche en Informatique et en Automatique (INRIA), March 9, 2010. URL http://hal.inria.fr/docs/00/46/24/81/PDF/RR-7215.pdf.
72. Thomas Weise, Raymond Chiong, Ke Tang, Jörg Lässig, Shigeyoshi Tsutsui, Wenxiang Chen, Zbigniew Michalewicz, and Xin Yao. Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine (CIM)*, 9(3):40–52, August 2014. doi: 10.1109/MCI.2014.2326101. URL http://www.it-weise.de/documents/files/WCTLTCMY2014BOAAOSFFTTSP.pdf. Featured article and selected paper at the website of the IEEE Computational Intelligence Society (http://cis.ieee.org/).

73. Mark S. Boddy and Thomas L. Dean. Solving time-dependent planning problems. Technical Report CS-89-03, Providence, RI, USA: Brown University, Department of Computer Science, February 1989. URL ftp://ftp.cs.brown.edu/pub/techreports/89/cs89-03.pdf.
74. John D. C. Little, Katta G. Murty, Dura W. Sweeny, and Caroline Karel. An algorithm for the traveling salesman problem. Sloan Working Papers 07-63, Cambridge, MA, USA: Massachusetts Institute of Technology (MIT), Sloan School of Management, March 1, 1963. URL http://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf.
75. Weixiong Zhang. Truncated branch-and-bound: A case study on the asymmetric traveling salesman problem. In *Proceedings of the AAAI-93 Spring Symposium on AI and NP-Hard Problems*, pages 160–166, Stanford, CA, USA, 1993. Menlo Park, CA, USA: AAAI Press. URL www.cs.wustl.edu/~zhang/publications/atsp-aaai93-symp.ps.
76. Weixiong Zhang. Truncated and anytime depth-first branch and bound: A case study on the asymmetric traveling salesman problem. In Weixiong Zhang and Sven König, editors, *AAAI Spring Symposium Series: Search Techniques for Problem Solving Under Uncertainty and Incomplete Information*, volume SS-99-07 of *AAAI Technical Report*, pages 148–155. Menlo Park, CA, USA: AAAI Press, 1999. URL https://www.aaai.org/Papers/Symposia/Spring/1999/SS-99-07/SS99-07-026.pdf.
77. Gerhard Reinelt. Tsplib – a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991. doi: 10.1287/ijoc.3.4.376.
78. Gerhard Reinelt. Tsplib 95. Technical report, Heidelberg, Germany: Universität Heidelberg, Institut für Mathematik, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/DOC.PS.
79. Gerhard Reinelt. Tsplib, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
80. Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking: Experimental setup. Technical report, Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, March 24, 2012. URL http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocexperiment.pdf.
81. Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Rapports de Recherche RR-6828, Institut National de Recherche en Informatique et en Automatique (INRIA), October 16, 2009. URL http://hal.archives-ouvertes.fr/inria-00362649/en/. Version 3.

82. Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions. Technical report, April 13, 2013. URL http://coco.lri.fr/downloads/download13.09/bbobdocfunctions.pdf. Working Paper 2009/20, compiled April 13, 2013.
83. Yan Jiang, Thomas Weise, Jörg Lässig, Raymond Chiong, and Rukshan Athauda. Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the tsp. In *Proceedings of the IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS'14), Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI'14)*, Orlando, FL, USA: Caribe Royale All-Suite Hotel and Convention Center, 2014. Los Alamitos, CA, USA: IEEE Computer Society Press. URL http://www.it-weise.de/documents/files/JWLCA2014CAHBABAWECMLSATHOTT.pdf.
84. Holger H. Hoos and Thomas Stützle. Evaluating las vegas algorithms – pitfalls and remedies. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 238–245, Madison, WI, USA, 1998. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://www.intellektik.informatik.tu-darmstadt.de/TR/1998/98-02.ps.Z. Also published as Technical Report "Forschungsbericht AIDA-98-02" of the Fachgebiet Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt, Germany.
85. *Encapsulated PostScript File Format Specification*. Number Tech Note #5002. Version 3.0 edition, May 1, 1992. URL http://partners.adobe.com/public/developer/en/ps/5002.EPSF_Spec.pdf.
86. *Document Management – Portable Document Format – Part 1: PDF 1.7*. Number ISO 32000-1:2008. July 2008.
87. Thomas Boutell, et al., and USA: Boutell.Com Inc. Philadelphia, PA. *PNG (Portable Network Graphics) Specification Version 1.0*, volume 2083 of *Request for Comments (RFC)*. Network Working Group, March 1997. URL http://tools.ietf.org/html/rfc2083.
88. USA: CompuServe Incorporated Columbus, OH. Graphics interchange format(sm), version 89a, programming reference, July 31, 1990. URL http://www.w3.org/Graphics/GIF/spec-gif89a.txt.
89. Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion*. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 2004. ISBN 0-201-36299-6.
90. Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 1994. ISBN 0201541998 and 9780201541991. URL http://books.google.de/books?id=54A3MuBzIrEC.

91. Leslie Lamport. *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 1994. ISBN 0201529831 and 9780201529838. URL http://books.google.de/books?id=19pzDwEACAAJ.
92. Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX2ϵ – Or LaTeX2ϵ in 157 minutes*. 5.01 edition, April 6, 2011. URL http://tobi.oetiker.ch/lshort/lshort.pdf.
93. Sebastian Rahtz, Akira Kakuto, Karl Berry, Manuel Pégourié-Gonnard, Norbert Preining, Peter Breitenlohner, Reinhard Kotucha, Siep Kroonenberg, Staszek Wawrykiewicz, and Tomasz Trzeciak. *TeX Live*. Portland, OR, USA: TeX Users Group (TUG), June 30, 2013. URL http://www.tug.org/texlive/.
94. Christian Schenk. *MiKTEX ... typesetting beautiful documents...*. 2013. URL http://miktex.org/.
95. Gerald Murray, Silvano Balemi, Jon Dixon, Peter Nüchter, Jürgen von Hagen, and Michael Shell. Official ieee latex class for authors of the institute of electrical and electronics engineers (ieee) transactions journals and conferences, May 3, 2007. URL http://www.michaelshell.org/tex/ieeetran/.
96. Llncs document class – springer verlag latex2e support for lecture notes in computer science, June 12, 2010. URL ftp://ftp.springer.de/pub/tex/latex/llncs/latex2e/llncs2e.zip.
97. Gerald Murray and G.K.M. Tobin. Sig-alternate.cls – version 2.4 (compatible with the acm_proc_article-sp.cls" v3.2sp), April 22, 2009. URL http://www.acm.org/sigs/publications/proceedings-templates.
98. Murray Altheim and Shane McCarron. *XHTML™ 1.1 – Module-based XHTML – Second Edition*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), November 23, 2010. URL http://www.w3.org/TR/2010/REC-xhtml11-20101123.
99. Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. An evaluation of sequential model-based optimization for expensive blackbox functions. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1209–1216, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2501592. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0311-hutter.pdf.
100. Tianjun Liao and Thomas Stützle. Bounding the population size of ipop-cma-es on the noiseless bbob testbed. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1161–1168, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2482694. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0304-liao.pdf.

101. Tianjun Liao and Thomas Stützle. Testing the impact of parameter tuning on a variant of ipop-cma-es with a bounded maximum population size on the noiseless bbob testbed. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1169–1176, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2482695. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0305-liao.pdf.

102. Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julián Merelo-Guervós, and Hans-Paul Schwefel, editors, *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917/2000 of *Lecture Notes in Computer Science (LNCS)*, pages 849–858, Paris, France, 2000. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45356-3_83. URL https://eprints.kfupm.edu.sa/17643/1/17643.pdf.

103. Thanh-Do Tran, Dimo Brockhoff, and Bilel Derbel. Multiobjectivization with nsga-ii on the noiseless bbob testbed. In Christian Blum and Enrique Alba Torres, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1217–1224, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2482700. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0312-tran.pdf.

104. László Pál. Benchmarking a hybrid multi level single linkage algorithm on the bbob noiseless testbed. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1145–1152, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0302-pal.pdf.

105. László Pál. Comparison of multistart global optimization algorithms on the bbob noiseless testbed. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1153–1160, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2482693. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0303-pal.pdf.

106. John Ashworth Nelder and Roger A. Mead. A simplex method for function minimization. *The Computer Journal, Oxford Journals*, 7(4):308–313, January 1965. doi: 10.1093/comjnl/7.4.308. URL http://www.rupley.com/~jar/Rupley/Code/src/simplex/nelder-mead-simplex.pdf.

107. Neal J. Holtschulte and Melanie Moses. Benchmarking cellular genetic algorithms on the bbob noiseless testbed. In Christian Blum and Enrique Alba Torres, editors, *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1201–1208, Amsterdam, The Netherlands, 2013. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2464576.2482699. URL http://coco.gforge.inria.fr/lib/exe/fetch.php?media=pdf2013:w0309-holtschulte.pdf.
108. Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Proceedings of the 11th International Conference on Parallel Problem Solving From Nature, Part 2 (PPSN'10-2)*, volume 6239 of *Lecture Notes in Computer Science (LNCS)*, pages 300–309, Kraków, Poland: AGH University of Science and Technology, 2010. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-642-15871-1_31. URL http://www.it-weise.de/documents/files/CWYT2010LSGOUCCWVIL.pdf.
109. Scott Chacon and Ben Straub. *Pro Git: Everything you need to know about Git*. New York, NY, USA: Apress, Inc., 2nd edition, 2014. URL http://www.git-scm.com/book/en/v2.
110. Chris Dawson and Timothy M. O'Brien. *Github: Amplify your Software Development with Social Coding*. Sebastopol, CA, USA: O'Reilly Media, Inc., 1st edition, October 25, 2015. ISBN 1449368018 and 978-1449368012.
111. Richard E. Silverman. *Git Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., August 2, 2013. ISBN 1449325866 and 978-1449325862.
112. *Eclipse*. Ottawa, ON, Canada: Eclipse Foundation. URL http://www.eclipse.org/.
113. Brian R. Jackson. *Maven: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2nd edition, December 25, 2015. ISBN 144936280X and 978-1449362805.
114. Balaji Varanasi and Sudha Belida. *Introducing Maven*. New York, NY, USA: Apress, Inc., November 26, 2014. ISBN 1484208420 and 978-1484208427.
115. Kent Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009. ISBN 1449379028 and 9781449379025. URL http://books.google.de/books?id=Ur_zMK0WQwIC.
116. Vincent Massol and Ted Husted. *Junit In Action*. Greenwich, CT, USA: Manning Publications Co., 2004. ISBN 8177225383 and 9788177225389. URL http://books.google.de/books?id=P1mDmZUmje0C.
117. Joe B. Rainsberger and Scott Stirling. *Junit Recipes: Practical Methods for Programmer Testing*. Manning Pubs Co. Greenwich, CT, USA: Manning Publications Co., 2005. ISBN 1932394230 and 9781932394238. URL http://books.google.de/books?id=5h7oDjuY5WYC.

- In the `optimizationBenchmarking` project, we follow a distributed, concurrent software development process

- In the `optimizationBenchmarking` project, we follow a distributed, concurrent software development process
- We use `git` [109] as versioning system

- In the `optimizationBenchmarking` project, we follow a distributed, concurrent software development process
- We use `git` [109] as versioning system and `gitHub` [109–111] for hosting

- In the `optimizationBenchmarking` project, we follow a distributed, concurrent software development process
- We use `git` [109] as versioning system and `gitHub` [109–111] for hosting
- For building and dependency management, we use `Maven`

- In the optimizationBenchmarking project, we follow a distributed, concurrent software development process
- We use git [109] as versioning system and gitHub [109–111] for hosting
- For building and dependency management, we use Maven
- As developer environment, we recomment Eclipse [112] (version $\geq$ Luna), as it natively supports git and Maven [113, 114].

1. Prerequisites

1. Prerequisites:
   1. Obtain a `gitHub` account

1. Prerequisites:
   1. Obtain a `gitHub` account
   2. Register a public/private key pair for your account

Contribution Lifecycle

1. Prerequisites:
    1. Obtain a `gitHub` account
    2. Register a public/private key pair for your account
    3. Join group `optimizationBenchmarking`

Contribution Lifecycle

1. Prerequisites
2. Fork project
   `optimizationBenchmarking/optimizationBenchmarking`

1. Prerequisites
2. Fork project
3. Add your code, e.g., an own evaluation module, in the appropriate location (maybe an own package)

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code
    - add `JUnit` [115–117] tests if possible

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code
   - add `JUnit` [115–117] tests if possible
   - provide examples, example data, and expected results

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code
5. Make sure your code is properly documented and that your commits contain sufficient explanations

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code
5. Make sure your code is properly documented
6. Create a `pull request`, i.e., ask me to include your code in the main project

Contribution Lifecycle

1. Prerequisites
2. Fork project
3. Add your code
4. Test your code
5. Make sure your code is properly documented
6. Create a `pull request`
7. After a discussion, your code will (very likely) become part of the main project

- Importing a project (or fork) from `gitHub` into `Eclipse` means to clone it to a local repository and then to work on that repository.

- Importing a project (or fork) from `gitHub` into `Eclipse` means to clone it to a local repository and then to work on that repository.
- Although `gitHub` offers cloning via HTTPS as the default, for me it worked better with SSH.

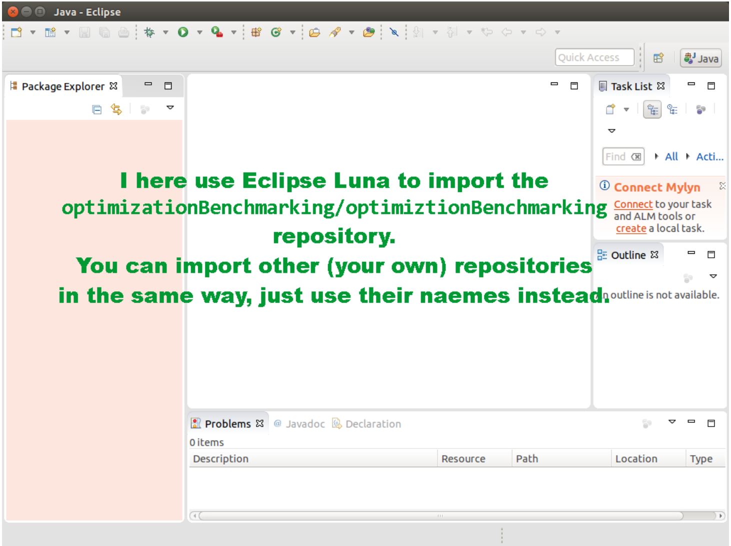
Import Fork into Eclipse

- Importing a project (or fork) from `gitHub` into `Eclipse` means to clone it to a local repository and then to work on that repository.
- Although `gitHub` offers cloning via HTTPS as the default, for me it worked better with SSH.
- After cloning and importing the clone into `Eclipse`, you need to update the project with `Maven` to properly initialize the project structure and dependencies.

Import Fork into Eclipse

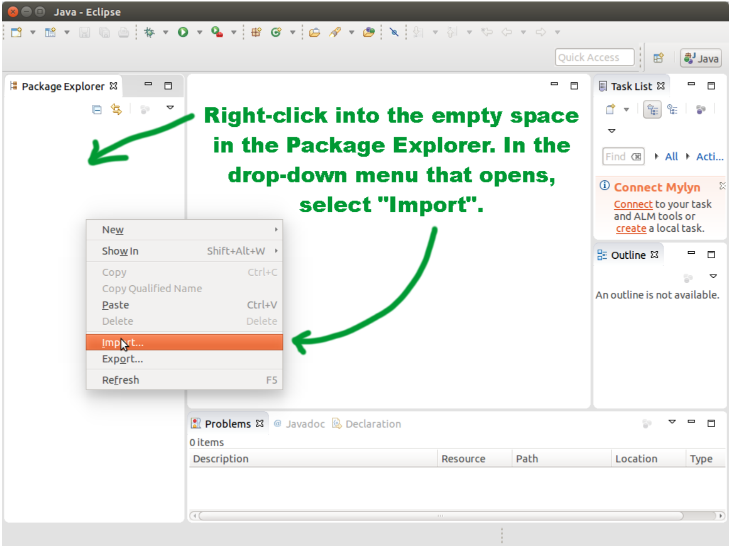
- Importing a project (or fork) from `gitHub` into `Eclipse` means to clone it to a local repository and then to work on that repository.
- Although `gitHub` offers cloning via HTTPS as the default, for me it worked better with SSH.
- After cloning and importing the clone into `Eclipse`, you need to update the project with `Maven` to properly initialize the project structure and dependencies.
- In the following, I provide a step-by-step screenshot series on how to do all of that...

Import Fork into Eclipse

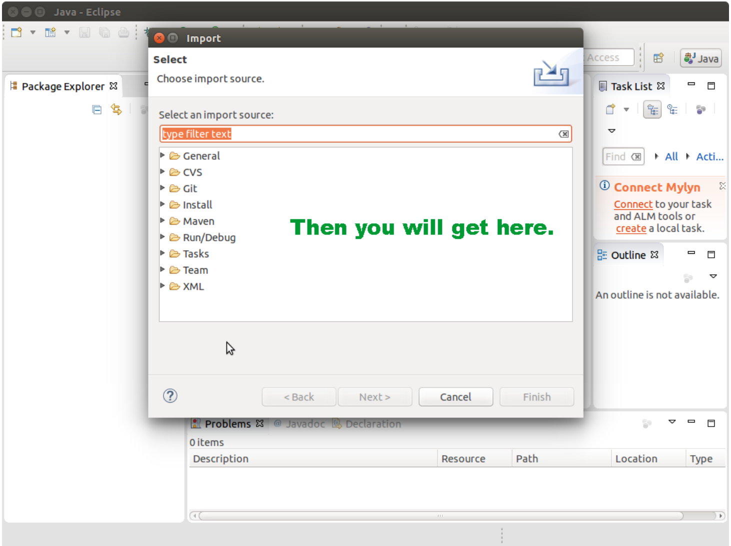


I here use Eclipse Luna to import the optimizationBenchmarking/optimiztionBenchmarking repository.

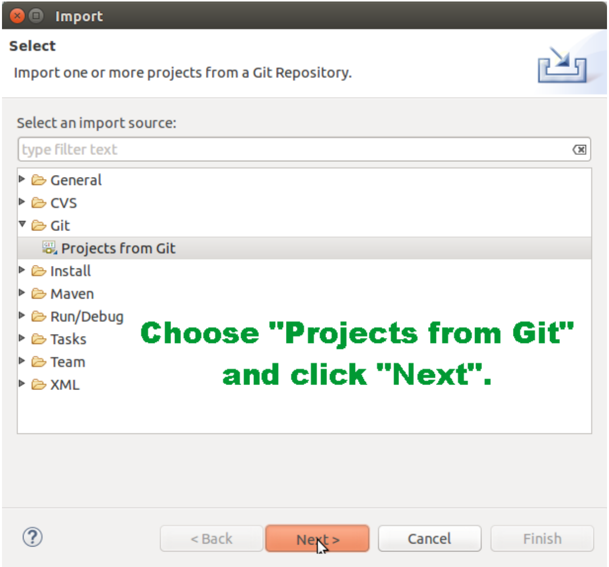
You can import other (your own) repositories in the same way, just use their naemes instead.


Java - Eclipse
Quick Access
Java
Package Explorer
Task List
Find
All
Acti...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Right-click into the empty space in the Package Explorer. In the drop-down menu that opens, select "Import".
New
Show In Shift+Alt+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Import...
Export...
Refresh F5
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

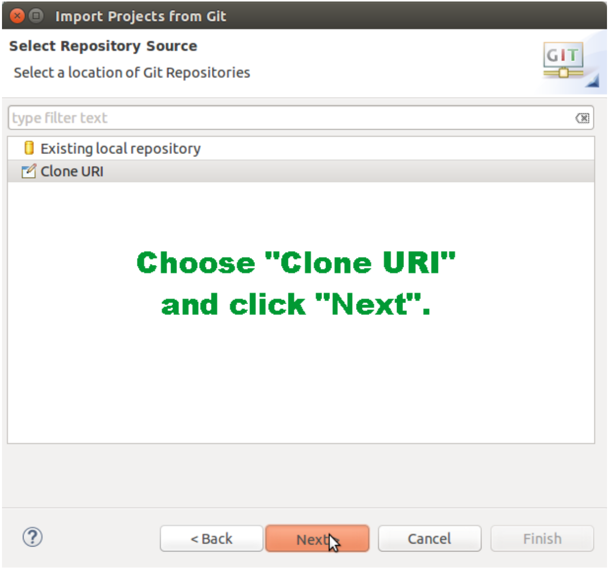
Import Fork into Eclipse

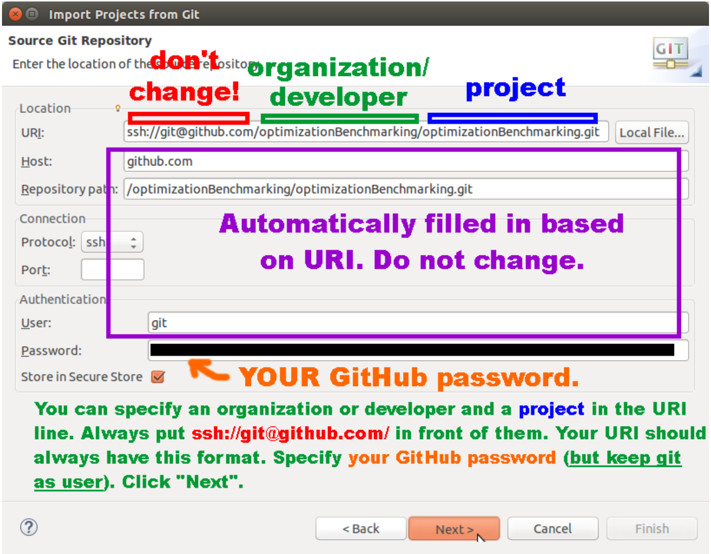


Import Fork into Eclipse



Import Fork into Eclipse

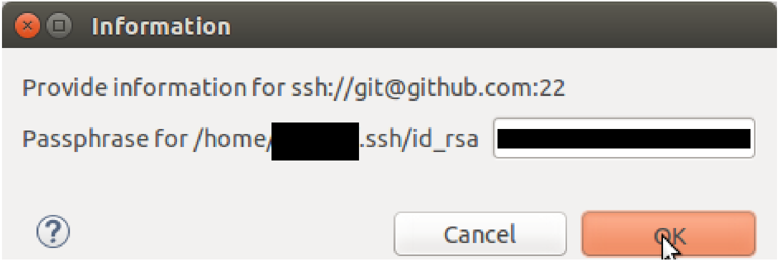




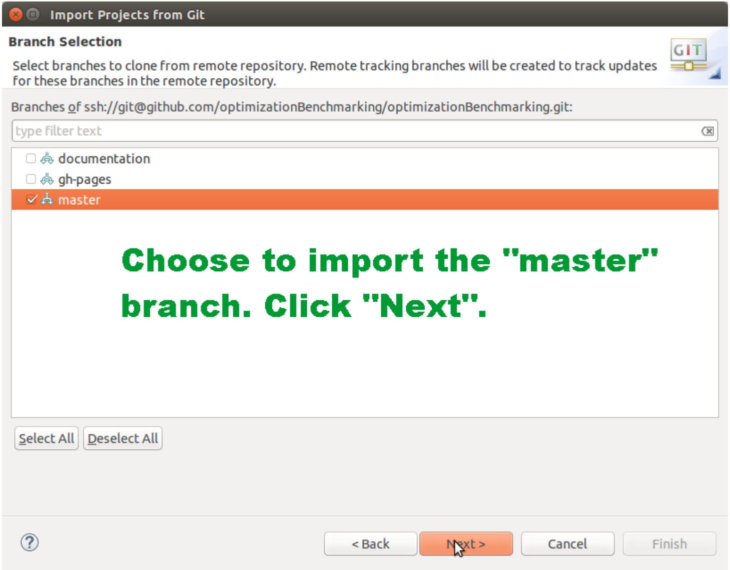
You can specify an organization or developer and a project in the URI line. Always put ssh://git@github.com/ in front of them. Your URI should always have this format. Specify your GitHub password (but keep git as user). Click "Next".

Import Fork into Eclipse

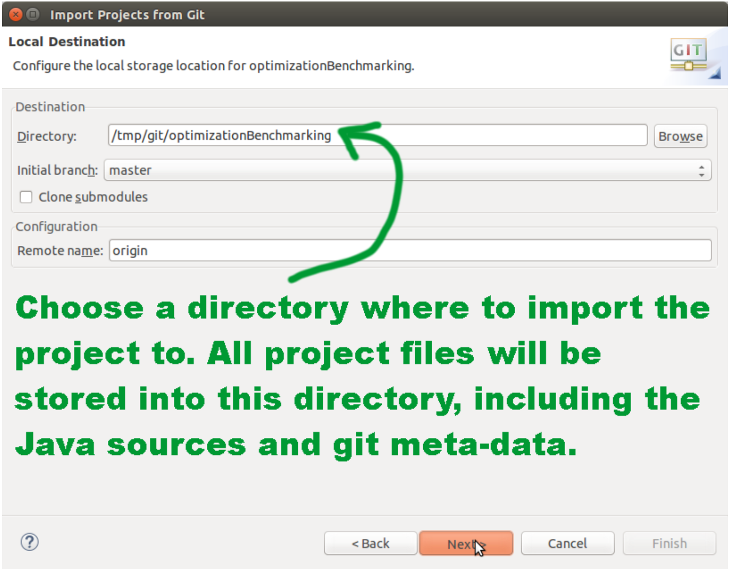
Provide your passphrase for your public/private key pair that you specified to GitHub.



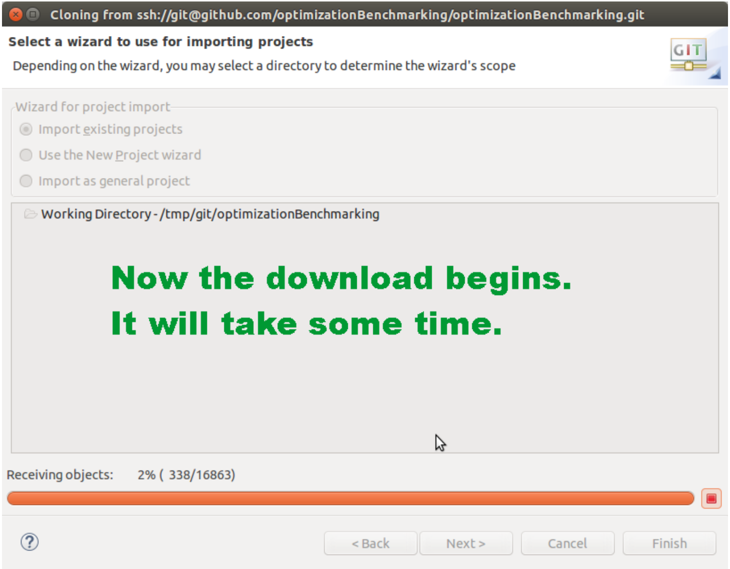
Import Fork into Eclipse

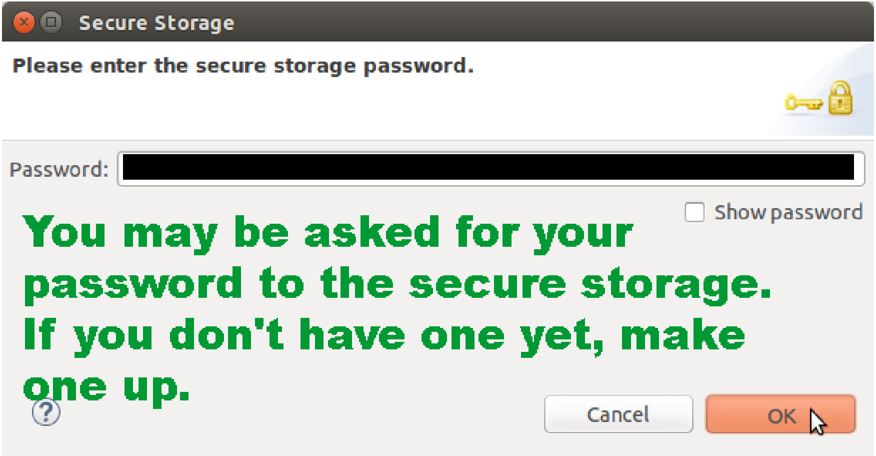


Import Fork into Eclipse

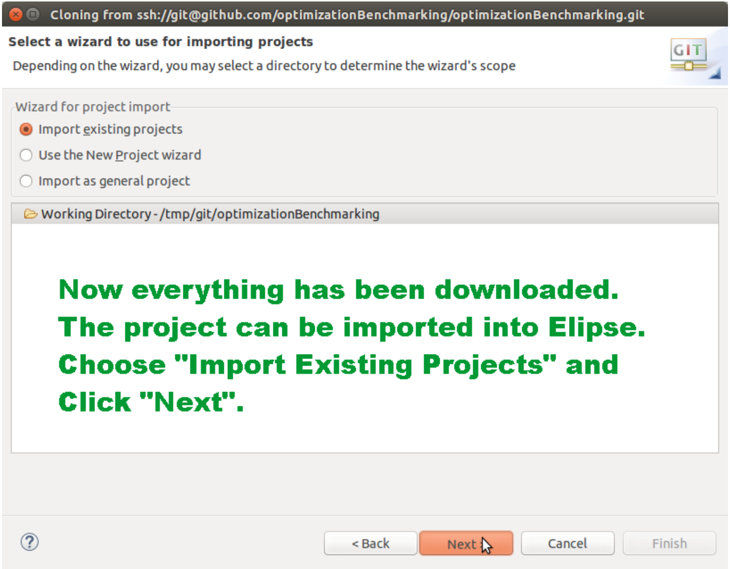


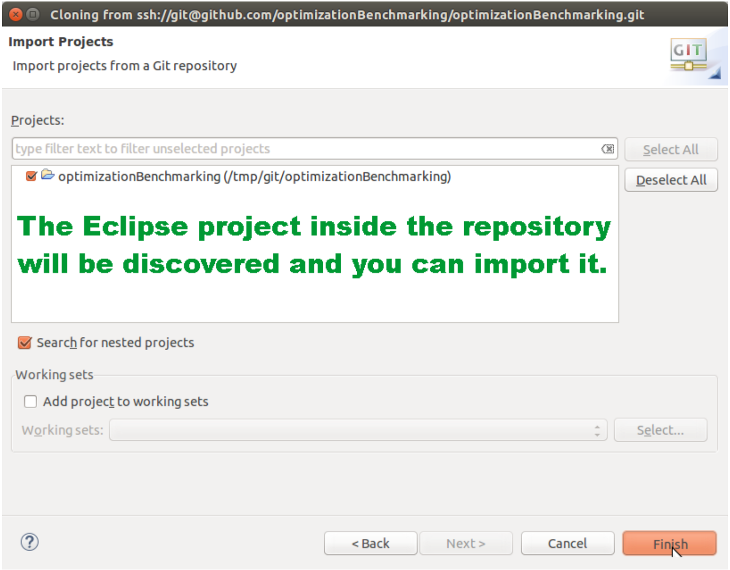
Import Fork into Eclipse



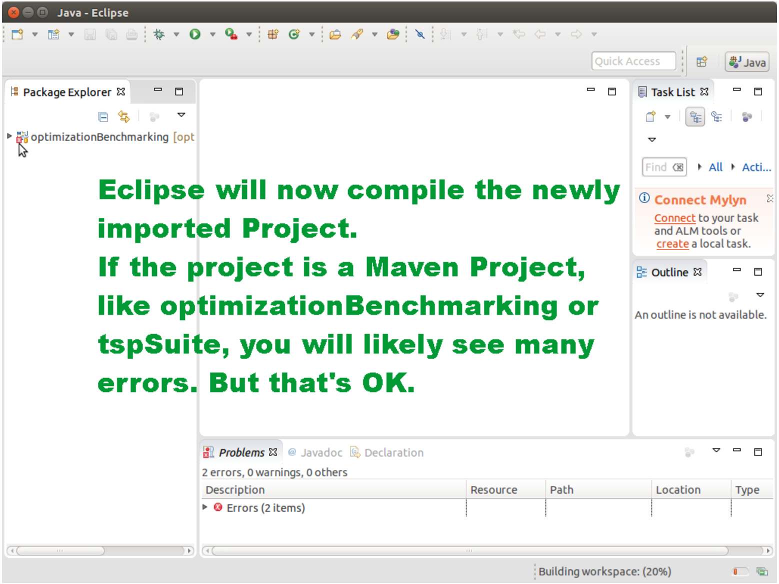

Secure Storage
Please enter the secure storage password.
Password:
Show password
You may be asked for your password to the secure storage. If you don't have one yet, make one up.
Cancel
OK

Import Fork into Eclipse

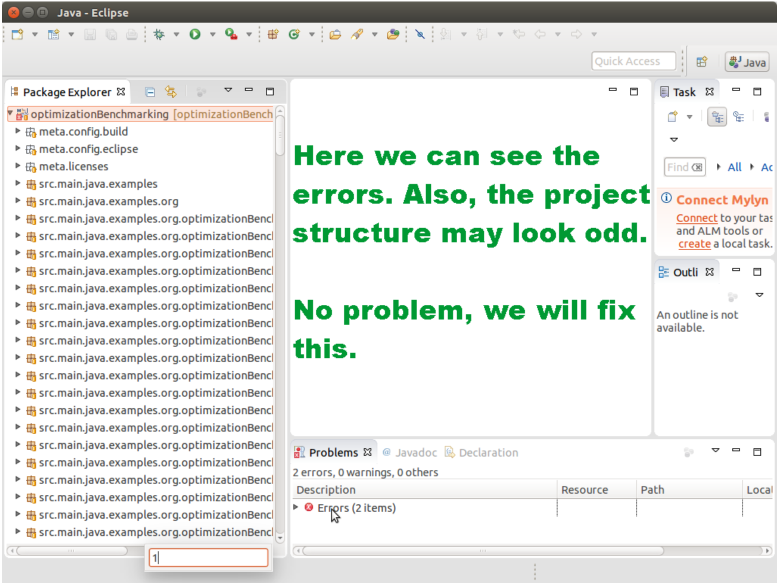



Cloning from ssh://git@github.com/optimizationBenchmarking/optimizationBenchmarking.git
Import Projects
Import projects from a Git repository
Projects:
type filter text to filter unselected projects
Select All
Deselect All
optimizationBenchmarking (/tmp/git/optimizationBenchmarking)
The Eclipse project inside the repository will be discovered and you can import it.
Search for nested projects
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish

Import Fork into Eclipse

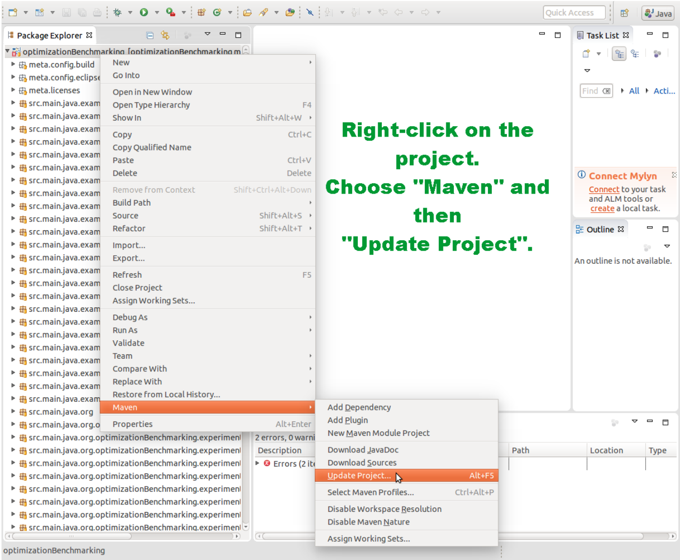


Eclipse will now compile the newly imported Project.
If the project is a Maven Project, like optimizationBenchmarking or tspSuite, you will likely see many errors. But that's OK.



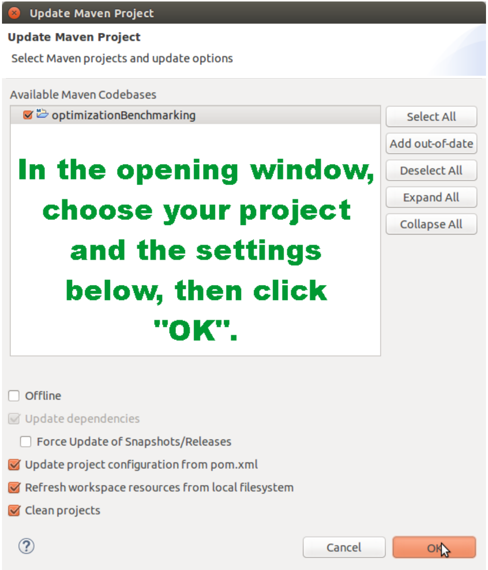
Here we can see the errors. Also, the project structure may look odd.

No problem, we will fix this.

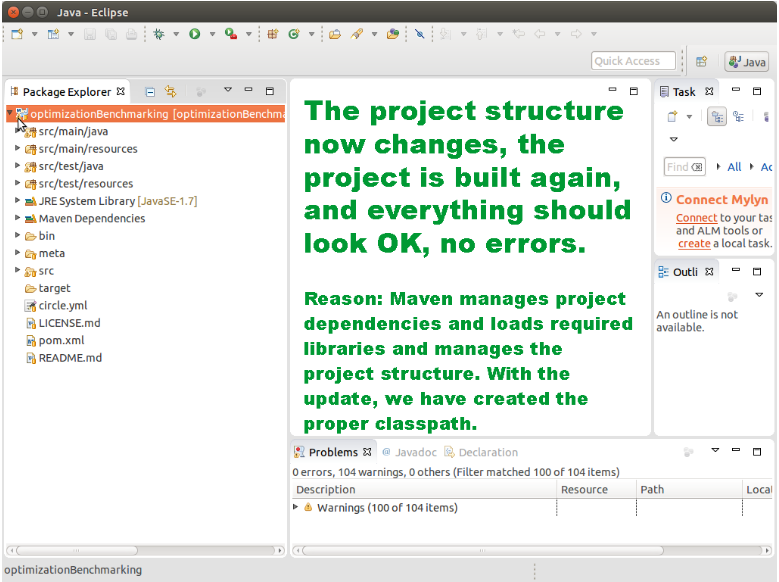


Right-click on the project. Choose "Maven" and then "Update Project".

Import Fork into Eclipse



Import Fork into Eclipse



The project structure now changes, the project is built again, and everything should look OK, no errors.

Reason: Maven manages project dependencies and loads required libraries and manages the project structure. With the update, we have created the proper classpath.

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- In *TSP Suite* [72, 83], we found a nice solution for that and *BBOB* [71, 80–82] follows a similar approach:

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- In *TSP Suite* [72, 83], we found a nice solution for that and *BBOB* [71, 80–82] follows a similar approach: Do everything in the objective function!

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function
  - it increases the internal FE counter by one

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function
  - it increases the internal FE counter by one
  - it checks whether a log point should be taken

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function
  - it increases the internal FE counter by one
  - it checks whether a log point should be taken
  - if so, it stores the log point in a pre-allocated memory location

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function
  - it increases the internal FE counter by one
  - it checks whether a log point should be taken
  - if so, it stores the log point in a pre-allocated memory location
  - it can store the objective value, the FE counter, and the ellapsed time

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`, which becomes `true`, e.g., when

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`, which becomes `true`, e.g., when
  - the FE counter reaches a certain maximum number

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`, which becomes `true`, e.g., when
  - the FE counter reaches a certain maximum number
  - the global optimum was found (which we know from `evaluate`)

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`, which becomes `true`, e.g., when
  - the FE counter reaches a certain maximum number
  - the global optimum was found (which we know from `evaluate`)
  - a certain time has ellapsed

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`
- After the run, all the log points held in memory are written to a file.

Log Points and Termination

- When benchmarking, the questions how to collect log points and when to terminate arises.
- Do everything in the objective function!
- The objective function loads the problem instance in its constructor
- It thus can provide information, like the number of clauses k or variables n in a MAX-SAT problem
- Whenever a candidate solution is evaluated via a provided `evaluate` function, a log point may be taken
- It also represents the termination criterion by providing a function `shouldTerminate`
- After the run, all the log points held in memory are written to a file. No file operations during the run to not mess up time measurements!

Visit our website

http://www.optimizationBenchmarking.org

or

http://optimizationbenchmarking.github.io/optimizationBenchmarking

for downloading the software (version 0.8.4) and obtaining more information.

System Requirements:

- Java 1.7 (Ideally a JDK, under JRE slower with more memory requirements)
- optional: a LaTeX installation, such as TeXLive or MiKTeX (needed for generating pdf reports)