Diese Maßnahme wird mitfinanziert durch Steuermittel auf Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Algorithm Benchmarking

Automating Research Work in Optimization

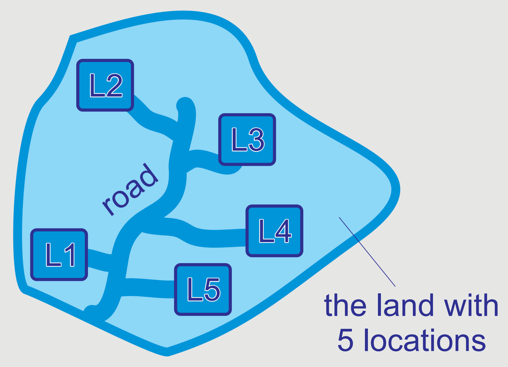
Thomas Weise[1], Abhishek Awasthi[2], Markus Ullrich[2], Jörg Lässig[2]

[1]Hefei University, Institute of Applied Optimization, Hefei, Anhui, China

[2]University of Applied Sciences Zittau/Görlitz, Enterprise Application Development Group, Görlitz, Germany

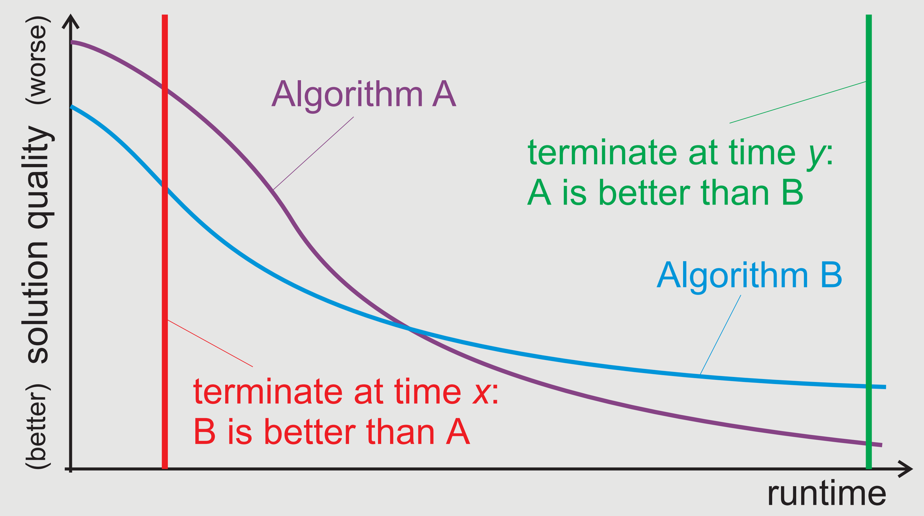
Optimization Algorithms

- many real world questions are optimization problems e.g.:
  - find the shortest path for a traveling salesman
  - optimize factory locations to minimize material transportation



Algorithm Performance

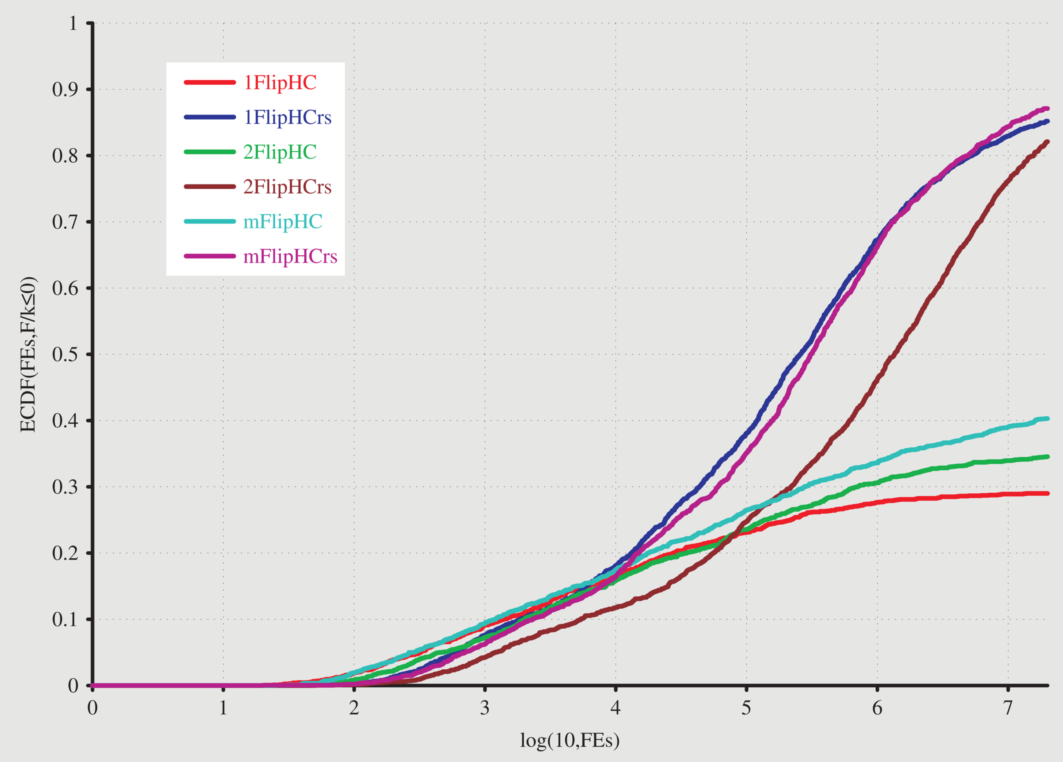
- has two dimensions: solution quality and required runtime
- anytime algorithms maintain an approximate solution:
  - at any time during their run **and**
  - iteratively improve this guess



Experimental Procedure

Research Questions

- Which optimization algorithm is best for my problem?
- An optimization algorithm can have parameters … which parameter settings make it work best?
- For a problem, there can be many concrete instances … which features make them hard or easy?
- Are there groups of algorithms (or problem instances) that behave differently? Why?
- How can I share problem instances or generate them reproducibly?



Example Result: plot of the Empirical (Cumulative) Distribute Function (ECDF), i.e. the fraction of runs that have found the solution for their respective problem at a given point in time.

Methodology

1. Select a set of benchmark instances.
2. Run experiments and collect data, f.e.:

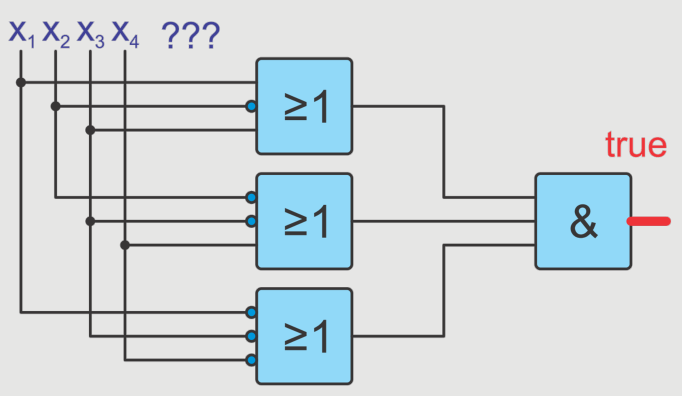
FEs	AT	NT	best obj. value
1	3	42.19	4075
2	5	70.32	3976
…			
24099	11393	160237.03	2579

3. Draw diagrams, print tables.
4. Identify interesting information, find reasons, go back to step 1.

Example Problem

Problem Description

- Maximum Satisfiability Problems (MAX-SAT):
  - Given: Formula B in Boolean logic with n Boolean variables $\vec{x} = (x_1, x_2, \dots, x_n)$, which appear either directly or negated in k "or" clauses, which are all combined with one "and"
  - MAX-SAT Goal: minimize objective function $f(\vec{x})$ = number of clauses which are `false`.
  - $f(\vec{x}) = 0 \Longrightarrow$ all clauses are `true`, SAT problem solved



Instance Generation

- supports a common format
  - different data types (columns)
  - multiple data entries (rows)
  - dependencies
- uses a blueprint for the generation
  - human-readable
  - re-usable
  - reproducible results
  - generic and easy to extend

Listing 1: Max-SAT Example Configuration based on the CNF Standard

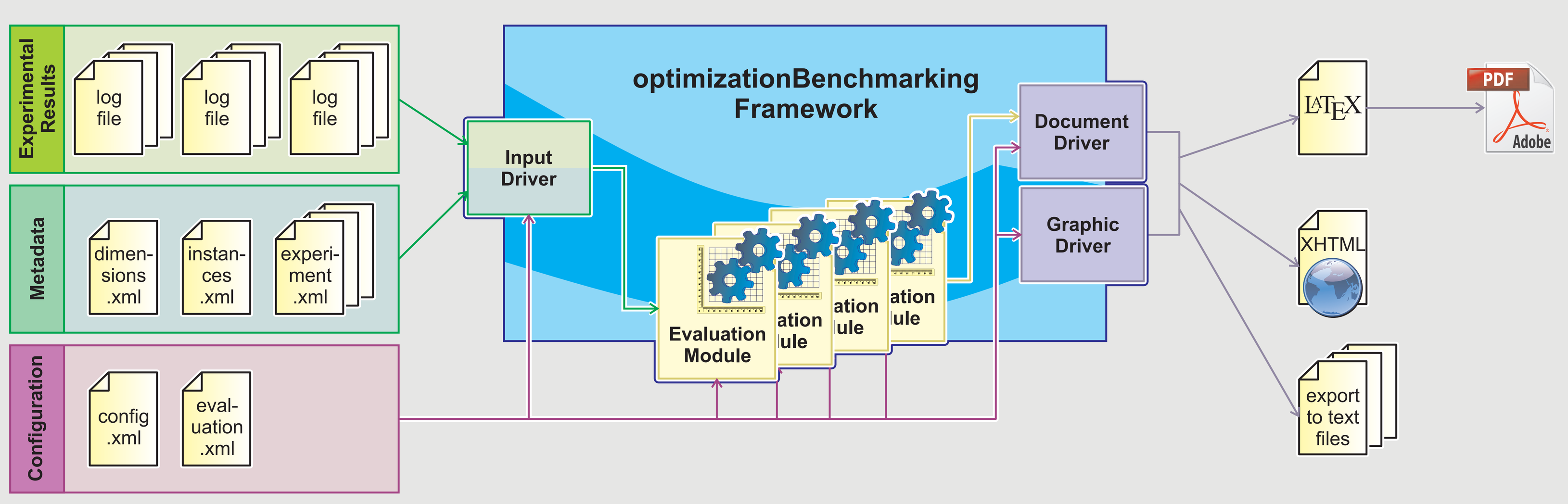
```
{  ...
  "separator": " ",
  "comment_prefix": "c",
  "alternative_header": "p cnf 4 100",
  "attributes": [
    { "name": "i",
      "type": "integer",
      "min": -4,
      "max": 4 },
    { "name": "j", ... },
    { "name": "k", ...,
      "output_probability": 0.3 }
    { "name": "zero",
      "type": "integer",
      "value": 0}
  ],
"constraints": [
    { "name": "i!=j",
      "left": {
        "type": "attribute",
        "value": "i" },
      "relation": "!=",
      "right": {
        "type": "attribute",
        "value": "j" } },
    { "name": "k!=j", ... },
    { "name": "k!=i", ... },
    { "name": "no_i_zero",
      "left": {
        "type": "attribute",
        "value": "i" },
      "relation": "!=",
      "right": {
        "type": "integer",
        "value": 0 } },
    { "name": "no_j_zero", ... },
    { "name": "no_k_zero", ... } }
  ]
}
```

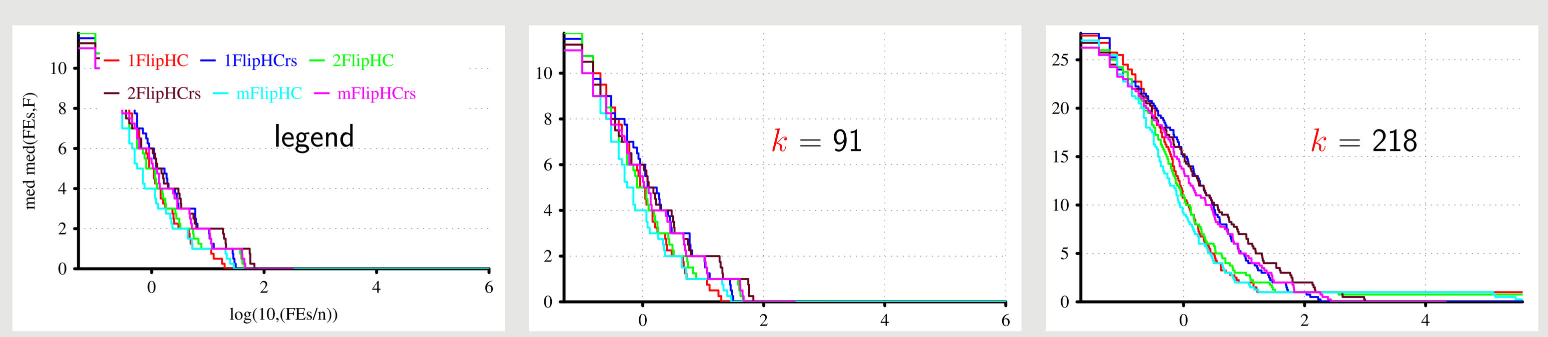
```
p cnf 4 100
-1 3 0
4 -2 0
-2 1 -3 0
...
```

$\Longleftarrow$

Data Analysis

- Goal: compare the performance of different algorithm setups.
  - load and evaluate the collected performance data,
  - understanding strengths and weaknesses of algorithms,
  - produce ready for use figures for publications. $\Longrightarrow$





Results for several hill climber algorithms on selected instances of the MAX-SAT problem with varying k.

Selected Literature

1. Thomas Weise, Xiaofeng Wang, Qi Qi, Bin Li, and Ke Tang. Automatically discovering clusters of algorithm and problem instance behaviors as well as their causes from experimental data, algorithm setups, and instance features. Applied Soft Computing Journal (ASOC), 73:366-382. December 2018.
2. Markus Ullrich, Thomas Weise, Abhishek Awasthi, and Jörg Lässig. A Generic Problem Instance Generator for Discrete Optimization Problems, In BB-DOB Workshop at The Genetic and Evolutionary Computation Conference (GECCO'18).
3. Abhishek Awasthi, Jörg Lässig, Thomas Weise, and Oliver Kramer. Tackling Common Due Window Problem with a Two-Layered Approach. In Proceedings of the 10th International Conference on Combinatorial Optimization and Applications (COCOA 2016).
4. Thomas Weise, Yuezhong Wu, Raymond Chiong, Ke Tang, and Jörg Lässig. Global versus local search: The impact of population sizes on evolutionary algorithm performance. Journal of Global Optimization, February 2016.
5. 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.

STUDIEREN_OHNE_GRENZEN

SMWK Landtagsmittel-Projektekonferenz, Zittau, Germany, November 7th, 2018