From Standardized Data Formats to Standardized Tools for Optimization Algorithm Benchmarking

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus	合肥学院 南艳湖校区
Faculty of Computer Science and Technology	计算机科学与技术系
Institute of Applied Optimization	应用优化研究所
230601 Hefei, Anhui, China	中国 安徽省 合肥市 230601
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99	经济技术开发区 锦绣大道99号

Outline

1. Introduction
2. Tools for Research on Optimization
3. Example Experiment and Data
4. Conclusions

website

Section Outline

1. Introduction
2. Tools for Research on Optimization
3. Example Experiment and Data
4. Conclusions

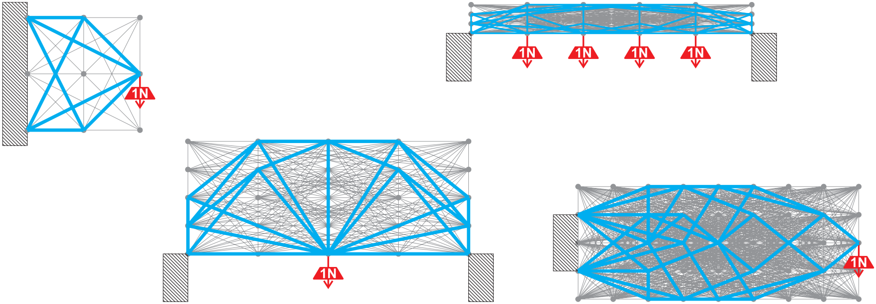
website

Optimization

- Many questions in the real world are *optimization problems*

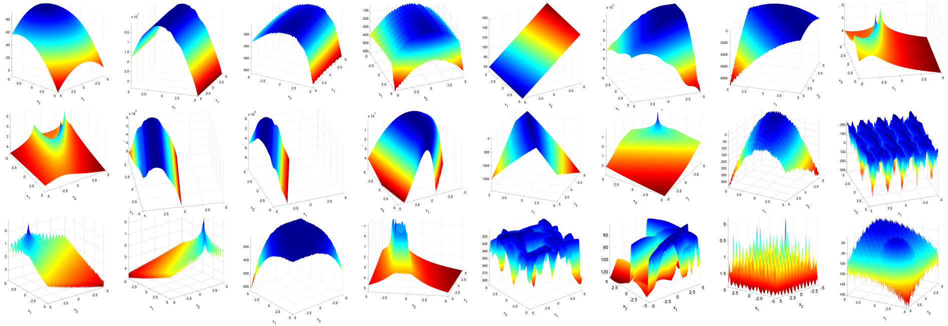
- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities in China and return to Hefei!

- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - How can I construct a truss which can hold a certain weight with at most a certain amount of iron?



Optimization

- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - Construct a truss which can hold a certain weight
  - Find the minima of complex, multi-dimensional mathematical formulas



"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]:

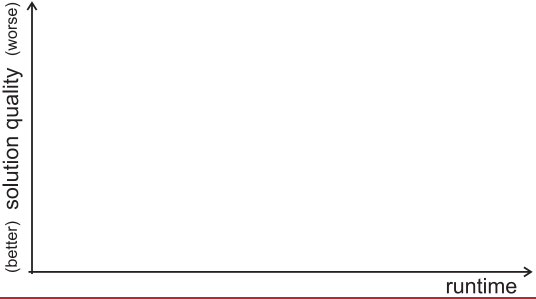
"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

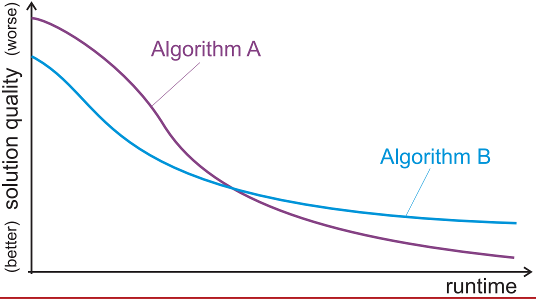
- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] 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 [4–6] are Anytime Algorithms.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] 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 [4–6] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] 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 [4–6] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.
- We can let them run arbitrarily long, there usually is no explicit, natural end point

Performance and Anytime Algorithms

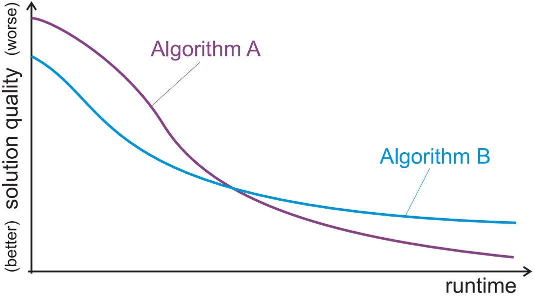
"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- Anytime Algorithms [3] 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 [4–6] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.
- We can let them run arbitrarily long, there usually is no explicit, natural end point
- Experiments must capture data on the whole runtime behavior!

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

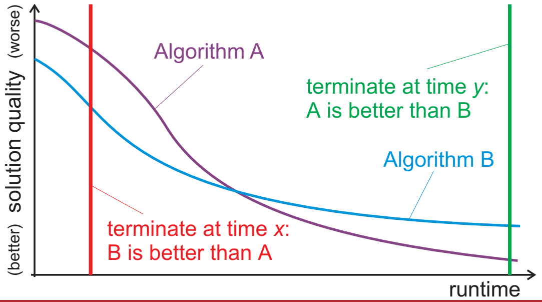
- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- If we just compare "final" results, we may arrive at incomplete conclusions



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

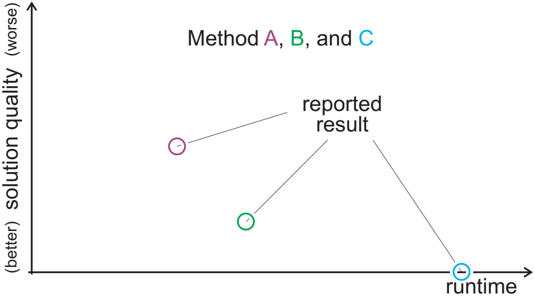
- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- If we just compare "final" results, we may arrive at incomplete conclusions



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

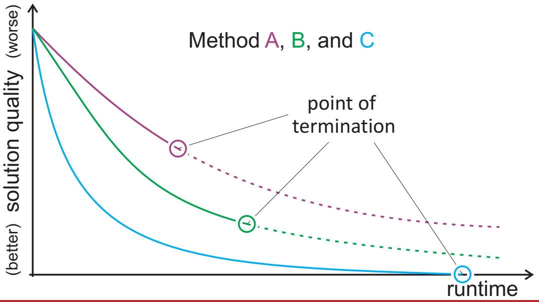
- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- If we just compare "final" results, we may arrive at incomplete or entirely wrong conclusions



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [1, 2]: solution quality and required runtime
- If we just compare "final" results, we may arrive at incomplete or entirely wrong conclusions



Section Outline

Research on Optimization

- What questions does research on optimization ask?

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.
  3. Draw diagrams, print tables (often summarizing over groups of instances or algorithms).

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.
  3. Draw diagrams, print tables (often summarizing over groups of instances or algorithms).
  4. Identify interesting information, find reasons, go back to step 1

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters . . . which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances . . . which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.
  3. Draw diagrams, print tables (often summarizing over groups of instances or algorithms).
  4. Identify interesting information, find reasons, go back to step 1
- This is a lot of work.

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.
  3. Draw diagrams, print tables (often summarizing over groups of instances or algorithms).
  4. Identify interesting information, find reasons, go back to step 1
- This is a lot of work. And much data is needed, due to anytime character of algorithms.

Research on Optimization

- What questions does research on optimization ask?
  - Which optimization algorithm is best for my problem?
  - An optimization algorithm can have parameters ... which parameter settings make it work the best?
  - For an optimization problem, there can be many concrete instances ... which features make them hard or easy?
- How do researchers answer these questions?
  1. Select (or develop) different algorithms/setups on different problem instances.
  2. Run experiments and collect data about the algorithm progress over runtime.
  3. Draw diagrams, print tables (often summarizing over groups of instances or algorithms).
  4. Identify interesting information, find reasons, go back to step 1
- This is a lot of work. And much data is needed, due to anytime character of algorithms. Tools automating the evaluation procedure are needed.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs" (due to the usually stochastic nature of algorithms).

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)" to capture whole runtime behavior (not just a single result/time point...).

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run: then we can draw summary diagrams over "similar" algorithm setups.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run.
  2. Meta-data on the features of the problem instances: then we can draw summary diagrams over "similar" instances.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run.
  2. Meta-data on the features of the problem instances.
- Today, automated evaluation tools only exist for dedicated problems and specific algorithm types.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run.
  2. Meta-data on the features of the problem instances.
- Today, automated evaluation tools only exist for dedicated problems and specific algorithm types.
- Reason: each tool only supports its own, very specific file format and assumes fixed, predefined benchmark instances.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent “runs”.
  2. For each run, we need several tuples of “(elapsed runtime, solution quality)”.
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run.
  2. Meta-data on the features of the problem instances.
- Today, automated evaluation tools only exist for dedicated problems and specific algorithm types.
- Reason: each tool only supports its own, very specific file format and assumes fixed, predefined benchmark instances.
- With common formats for the above data, tools that can deal with *arbitrary* algorithms on *arbitrary* problems can be developed.

Which information is needed?

- Which information is needed to plot runtime/performance diagrams?
  1. For each algorithm on each problem, we need several independent "runs".
  2. For each run, we need several tuples of "(elapsed runtime, solution quality)".
- Which information is needed to allow for automatic grouping of data?
  1. Meta-data on algorithm parameters for each run.
  2. Meta-data on the features of the problem instances.
- Today, automated evaluation tools only exist for dedicated problems and specific algorithm types.
- Reason: each tool only supports its own, very specific file format and assumes fixed, predefined benchmark instances.
- With common formats for the above data, tools that can deal with *arbitrary* algorithms on *arbitrary* problems can be developed.
- The *optimizationBenchmarking.org* is an example for such tools.

Requirements for Data Formats

- Common data formats must be

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable
- We define a data format for

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable
- We define a data format for
  - collected runtime/quality tuples from experiments $\Longrightarrow$ text: space-separated values

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable
- We define a data format for
  - collected runtime/quality tuples from experiments $\Longrightarrow$ text: space-separated values
  - information about measurement dimensions $\Longrightarrow$ XML

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable
- We define a data format for
  - collected runtime/quality tuples from experiments $\Longrightarrow$ text: space-separated values
  - information about measurement dimensions $\Longrightarrow$ XML
  - information about algorithm parameters $\Longrightarrow$ XML

Requirements for Data Formats

- Common data formats must be
  - very easy to read/write/parse/generate
  - human-readable and human-editable
- We define a data format for
  - collected runtime/quality tuples from experiments $\Longrightarrow$ text: space-separated values
  - information about measurement dimensions $\Longrightarrow$ XML
  - information about algorithm parameters $\Longrightarrow$ XML
  - information about problem instance featues $\Longrightarrow$ XML

Section Outline

Experiment on MAX-3SAT Problem

- We perform an example experiment on the MAX-3SAT [7] domain

- We perform an example experiment on the MAX-3SAT [7] domain
- We want to compare the performance of six different (trivial) algorithm setups differing in two parameters

Experiment on MAX-3SAT Problem

- We perform an example experiment on the MAX-3SAT [7] domain
- We want to compare the performance of six different (trivial) algorithm setups differing in two parameters
- We use ten groups with ten problem instances each from *SATLib* [8], differing in two features (number k of clauses, number of variables n)

Experiment on MAX-3SAT Problem

- We perform an example experiment on the MAX-3SAT [7] domain
- We want to compare the performance of six different (trivial) algorithm setups differing in two parameters
- We use ten groups with ten problem instances each from *SATLib* [8], differing in two features (number k of clauses, number of variables n)
- We do 20 runs for each instance $\times$ algorithm setup combination

Experiment on MAX-3SAT Problem

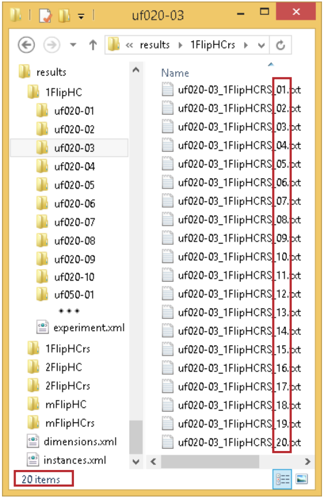
- We perform an example experiment on the MAX-3SAT [7] domain
- We want to compare the performance of six different (trivial) algorithm setups differing in two parameters
- We use ten groups with ten problem instances each from *SATLib* [8], differing in two features (number k of clauses, number of variables n)
- We do 20 runs for each instance $\times$ algorithm setup combination
- We prescribe this folder structure of `instance` $\longrightarrow$ `algorithm setup` $\longrightarrow$ `run(s).txt`, as it can be adopted for any kind experiment in optimization.

Obtained Data

- After the experiment. . .

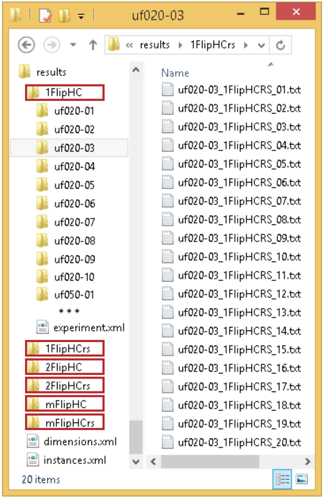
Obtained Data

- After the experiment…
  - …we have 20 independent runs (log files)

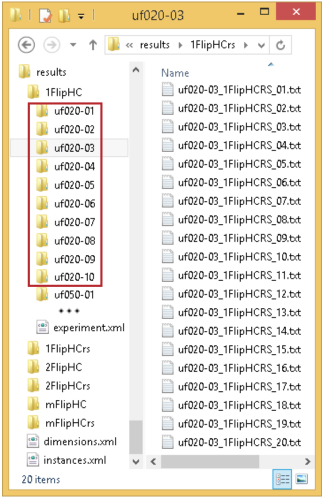


Obtained Data

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

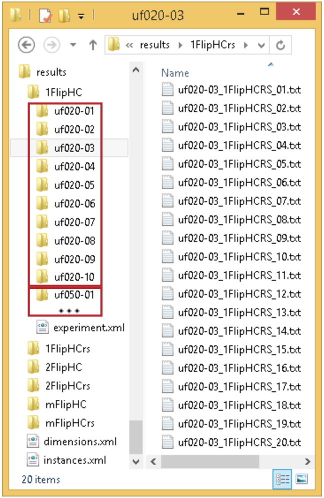


- After the experiment...
  - ...we have 20 independent runs (log files)
  - for each of the 6 algorithm setups,
  - on each of the 10 benchmark instances



Obtained Data

- 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



Example of Log File Structure

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2[nd] 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

iterations

```
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`.

```
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

iterations

runtime [ns]

Example of Log File Structure

- Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2[nd] 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

iterations

runtime [ns]

solution quality

Metadata

- Metadata is represented as XML.

Metadata

- Metadata on the measured dimensions is represented as XML.

Listing: The description of the measured dimensions.

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

  <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>
```

Metadata

- Metadata on the measured dimensions and the benchmark instance features is represented as XML.

Listing: The description of the benchmark instance features.

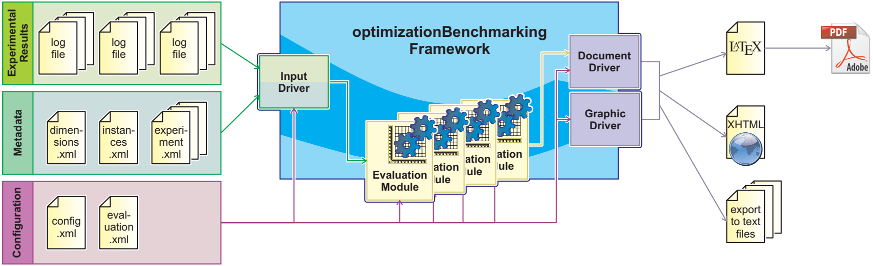
```
<?xml version="1.0" encoding="UTF-8"?>
<instances xmlns="http://www.optimizationBenchmarking.org/formats/...">
  <instance name="uf020-01"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 20 variables and 91 clauses.">
    <feature name="n" value="20" />
    <feature name="k" value="91" />
  </instance>
...
  <instance name="uf050-01"
    description="A uniformly randomly generated satisfiable 3-SAT instance with 50 variables and 218 clauses.">
    <feature name="n" value="50" />
    <feature name="k" value="218" />
  </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>
...
</instances>
```

Metadata

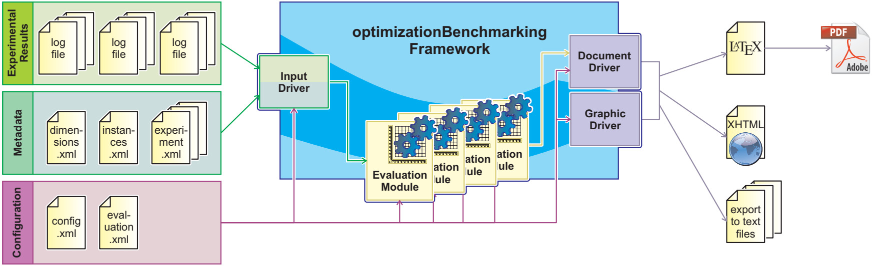
- Metadata on the measured dimensions, the benchmark instance features, and the algorithm setups is represented as XML.

Listing: The description of the parameters of one specific experiment setup.

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



- The *optimizationBenchmarking.org* framework is an example for software accepting data in such common formats.

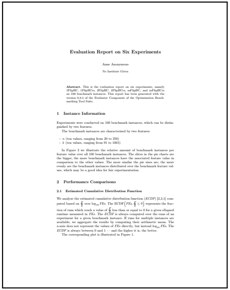


- The *optimizationBenchmarking.org* framework is an example for software accepting data in such common formats.
- It can be configured and launched via a web-based GUI and researchers can select, transform, and group data based on the meta-information.


Evaluation Report on Six Experiments

Anne Anonymous

first page of the report in LaTeX for IEEEtran


Evaluation Report on Six Experiments

Anne Anonymous

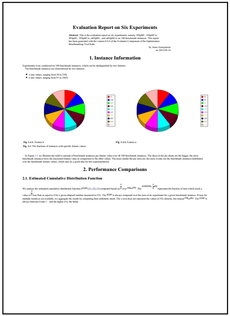
No Institute Given

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


Evaluation Report on Six Experiments

1. Instance Information

first page of the report in XHTML

- As result, it can produce human-readable reports with high-level conclusions and publication-ready diagrams from this data.

Conclusions

- Experimentation with optimization algorithms is complicated and involves much data if we want to do it right.

Conclusions

- Experimentation with optimization algorithms is complicated and involves much data if we want to do it right.
- We therefore need tool support.

Conclusions

- Experimentation with optimization algorithms is complicated and involves much data if we want to do it right.
- We therefore need tool support.
- The existing tool support is limited to specific problems, i.e., there is 1:1 relationship between tool and problem.

Conclusions

- Experimentation with optimization algorithms is complicated and involves much data if we want to do it right.
- We therefore need tool support.
- The existing tool support is limited to specific problems, i.e., there is 1:1 relationship between tool and problem.
- A general data format would lift this boundary, general tools could evolve.

Conclusions

- Experimentation with optimization algorithms is complicated and involves much data if we want to do it right.
- We therefore need tool support.
- The existing tool support is limited to specific problems, i.e., there is 1:1 relationship between tool and problem.
- A general data format would lift this boundary, general tools could evolve.
- We define such a format and give an example for a tool using it (*optimizationBenchmarking.org*).

谢谢
Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus
Institute of Applied Optimization
Hefei, Anhui, China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. 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`.
2. 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. Featured article and selected paper at the website of the IEEE Computational Intelligence Society (http://cis.ieee.org/).
3. 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`.
4. 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`.
5. 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`.
6. 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`.
7. 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`.
8. 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`.

Bibliography II

9. Thomas Weise. From standardized data formats to standardized tools for optimization algorithm benchmarking. In *Proceedings of the 16th IEEE Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC'17), July 26–28, 2017, University of Oxford, Oxford, UK*, Los Alamitos, CA, USA. IEEE Computer Society Press.

Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
    - Clearly, there is (at least) one shortest tour.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.

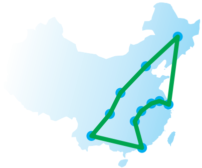
- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Theory proofs that the time to find this tour may grow exponentially with the number of cities we want to visit.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.

consumed runtime to find tour: very much / too(?) long

Heuristic Optimization

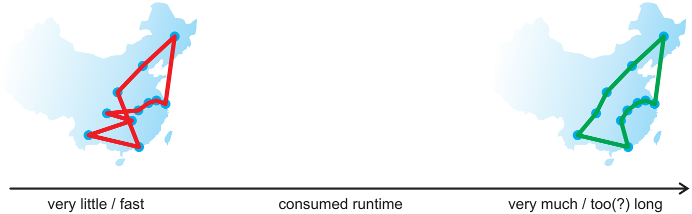
- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.



consumed runtime to find tour: very much / too(?) long

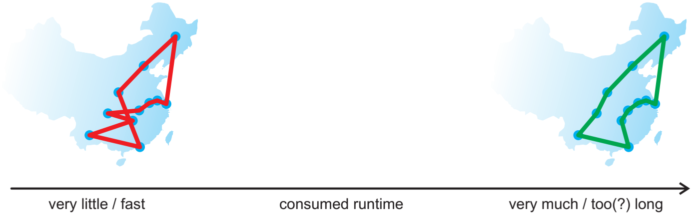
Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.

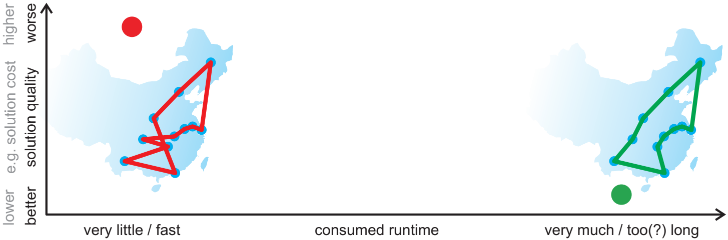


Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower: the tour will be longer than the best one.



- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower: the tour will be longer than the best one.



Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower.
  - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.