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
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 230601
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
1 Introduction

2 Tools for Research on Optimization

3 Example Experiment and Data

4 Conclusions
• 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?
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\textsuperscript{[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\cite{1,2}: solution quality
“(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
“(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.
“(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.
“(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.”

- Algorithm performance has two dimensions\cite{1, 2}: solution quality and required runtime
- Anytime Algorithms\cite{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\cite{4–6} are 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.
“(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.”

- Algorithm performance has two dimensions\textsuperscript{[1, 2]}: solution quality and required runtime.
- Anytime Algorithms\textsuperscript{[3]} are optimization methods which maintain an approximate solution at \textit{any time} during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound\textsuperscript{[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.
“(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!
“(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.

![Diagram](image-url)
"(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
“(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.”

- Algorithm performance has two dimensions \cite{1, 2}: solution quality and required runtime
- If we just compare “final” results, we may arrive at incomplete or entirely wrong conclusions

![Diagram](image-url)
"(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

![Graph illustrating the trade-off between solution quality and runtime for different methods (A, B, and C).](image)
Section Outline

1. Introduction
2. Tools for Research on Optimization
3. Example Experiment and Data
4. Conclusions
• What questions does research on optimization ask?
• 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?
• 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.
• 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).
• 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.
• 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 to plot runtime/performance diagrams?
• Which information is needed to plot runtime/performance diagrams?
  
  For each algorithm on each problem, we need several independent “runs” (due to the usually stochastic nature of algorithms).
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 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 optimizationBenchmarks.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 → 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 \(\Rightarrow\) text: space-separated values
  • information about measurement dimensions \(\Rightarrow\) XML

From Standardized Data Formats to Standardized Tools for Optimization Algorithm Benchmarking

Thomas Weise
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 → text: space-separated values
  - information about measurement dimensions → XML
  - information about algorithm parameters → 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 \(\rightarrow\) text: space-separated values
  - information about measurement dimensions \(\rightarrow\) XML
  - information about algorithm parameters \(\rightarrow\) XML
  - information about problem instance features \(\rightarrow\) XML
Section Outline

1 Introduction
2 Tools for Research on Optimization
3 Example Experiment and Data
4 Conclusions
• We perform an example experiment on the MAX-3SAT \cite{7} domain
• We perform an example experiment on the MAX-3SAT \cite{7} domain

• We want to compare the performance of six different (trivial) algorithm setups differing in two parameters
• We perform an example experiment on the MAX-3SAT \cite{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\cite{8}, differing in two features (number $k$ of clauses, number of variables $n$)
• We perform an example experiment on the MAX-3SAT \cite{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\cite{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\textsuperscript{[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 \textit{SATLib}\textsuperscript{[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 $\rightarrow$ algorithm setup $\rightarrow$ 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)
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
• 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\textsuperscript{nd} benchmark instance of set uf075.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9806</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24643</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>106040</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>115529</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>120373</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>144087</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>172967</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>1550118</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>296</td>
<td>1576034</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>297</td>
<td>1579525</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>1592492</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>323</td>
<td>1692189</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>332</td>
<td>1732127</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1082</td>
<td>5436999</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1558</td>
<td>7670059</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>9765759</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2024</td>
<td>9830168</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2809</td>
<td>13302012</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5246</td>
<td>24105640</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6330</td>
<td>28508740</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17284</td>
<td>73166926</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>60865</td>
<td>238968738</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Example of Log File Structure

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

<table>
<thead>
<tr>
<th>Log Point</th>
<th>Objective Value</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9806</td>
<td>46</td>
</tr>
<tr>
<td>3</td>
<td>24643</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>106040</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>115529</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>120373</td>
<td>21</td>
</tr>
<tr>
<td>25</td>
<td>144087</td>
<td>18</td>
</tr>
<tr>
<td>31</td>
<td>172967</td>
<td>16</td>
</tr>
<tr>
<td>290</td>
<td>1550118</td>
<td>15</td>
</tr>
<tr>
<td>296</td>
<td>1576034</td>
<td>14</td>
</tr>
<tr>
<td>297</td>
<td>1579525</td>
<td>13</td>
</tr>
<tr>
<td>300</td>
<td>1592492</td>
<td>12</td>
</tr>
<tr>
<td>323</td>
<td>1692189</td>
<td>10</td>
</tr>
<tr>
<td>332</td>
<td>1732127</td>
<td>9</td>
</tr>
<tr>
<td>1082</td>
<td>5436999</td>
<td>8</td>
</tr>
<tr>
<td>1558</td>
<td>7670059</td>
<td>7</td>
</tr>
<tr>
<td>2008</td>
<td>9765759</td>
<td>6</td>
</tr>
<tr>
<td>2024</td>
<td>9830168</td>
<td>5</td>
</tr>
<tr>
<td>2809</td>
<td>13302012</td>
<td>4</td>
</tr>
<tr>
<td>5246</td>
<td>24105640</td>
<td>3</td>
</tr>
<tr>
<td>6330</td>
<td>28508740</td>
<td>2</td>
</tr>
<tr>
<td>17284</td>
<td>73166926</td>
<td>1</td>
</tr>
<tr>
<td>60865</td>
<td>238968738</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Log File Structure

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

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

<table>
<thead>
<tr>
<th>log point</th>
<th>iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9806</td>
</tr>
<tr>
<td>3</td>
<td>24643</td>
</tr>
<tr>
<td>17</td>
<td>106040</td>
</tr>
<tr>
<td>19</td>
<td>115529</td>
</tr>
<tr>
<td>20</td>
<td>120373</td>
</tr>
<tr>
<td>25</td>
<td>144087</td>
</tr>
<tr>
<td>31</td>
<td>172967</td>
</tr>
<tr>
<td>290</td>
<td>1550118</td>
</tr>
<tr>
<td>296</td>
<td>1576034</td>
</tr>
<tr>
<td>297</td>
<td>1579525</td>
</tr>
<tr>
<td>300</td>
<td>1592492</td>
</tr>
<tr>
<td>323</td>
<td>1692189</td>
</tr>
<tr>
<td>332</td>
<td>1732127</td>
</tr>
<tr>
<td>1082</td>
<td>5436999</td>
</tr>
<tr>
<td>1558</td>
<td>7670059</td>
</tr>
<tr>
<td>2008</td>
<td>9765759</td>
</tr>
<tr>
<td>2024</td>
<td>9830168</td>
</tr>
<tr>
<td>2809</td>
<td>13302012</td>
</tr>
<tr>
<td>5246</td>
<td>24105640</td>
</tr>
<tr>
<td>6330</td>
<td>28508740</td>
</tr>
<tr>
<td>17284</td>
<td>73166926</td>
</tr>
<tr>
<td>60865</td>
<td>238968738</td>
</tr>
</tbody>
</table>
Example log file obtained from applying the 2-flip Hill Climber with Restarts to the 2\textsuperscript{nd} benchmark instance of set uf075.

<table>
<thead>
<tr>
<th>log point</th>
<th>iterations</th>
<th>runtime [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9806</td>
<td>46</td>
</tr>
<tr>
<td>3</td>
<td>24643</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>106040</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>115529</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>120373</td>
<td>21</td>
</tr>
<tr>
<td>25</td>
<td>144087</td>
<td>18</td>
</tr>
<tr>
<td>31</td>
<td>172967</td>
<td>16</td>
</tr>
<tr>
<td>290</td>
<td>1550118</td>
<td>15</td>
</tr>
<tr>
<td>296</td>
<td>1576034</td>
<td>14</td>
</tr>
<tr>
<td>297</td>
<td>1579525</td>
<td>13</td>
</tr>
<tr>
<td>300</td>
<td>1592492</td>
<td>12</td>
</tr>
<tr>
<td>323</td>
<td>1692189</td>
<td>10</td>
</tr>
<tr>
<td>332</td>
<td>1732127</td>
<td>9</td>
</tr>
<tr>
<td>1082</td>
<td>5436999</td>
<td>8</td>
</tr>
<tr>
<td>1558</td>
<td>7670059</td>
<td>7</td>
</tr>
<tr>
<td>2008</td>
<td>9765759</td>
<td>6</td>
</tr>
<tr>
<td>2024</td>
<td>9830168</td>
<td>5</td>
</tr>
<tr>
<td>2809</td>
<td>13302012</td>
<td>4</td>
</tr>
<tr>
<td>5246</td>
<td>24105640</td>
<td>3</td>
</tr>
<tr>
<td>6330</td>
<td>28508740</td>
<td>2</td>
</tr>
<tr>
<td>17284</td>
<td>73166926</td>
<td>1</td>
</tr>
<tr>
<td>60865</td>
<td>238968738</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of Log File Structure

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

<table>
<thead>
<tr>
<th>Log Point</th>
<th>Iterations</th>
<th>Runtime [ns]</th>
<th>Solution Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>9806</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>106040</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>115529</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>120373</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>144087</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>172967</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>290</td>
<td>1550118</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>296</td>
<td>1576034</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>297</td>
<td>1579525</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>300</td>
<td>1592492</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>323</td>
<td>1692189</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>332</td>
<td>1732127</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>1082</td>
<td>5436999</td>
<td>9</td>
</tr>
<tr>
<td>14</td>
<td>1558</td>
<td>7670059</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>2008</td>
<td>9765759</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>2024</td>
<td>9830168</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>2809</td>
<td>13302012</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>5246</td>
<td>24105640</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>6330</td>
<td>28508740</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>17284</td>
<td>73166926</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>60865</td>
<td>238968738</td>
<td>1</td>
</tr>
</tbody>
</table>

From Standardized Data Formats to Standardized Tools for Optimization Algorithm Benchmarking

Thomas Weise

13/22
• Metadata is represented as XML.
• Metadata on the measured dimensions is represented as XML.

Listing: The description of the measured dimensions.

```xml
<?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 on the measured dimensions and the benchmark instance features is represented as XML.

Listing: The description of the benchmark instance features.

```xml
<?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 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
<?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.
As a result, it can produce human-readable reports with high-level conclusions and publication-ready diagrams from this data.
1 Introduction

2 Tools for Research on Optimization

3 Example Experiment and Data

4 Conclusions
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).
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


• 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
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
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.
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.
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.