Algorithmen II

Peter Sanders, Christian Schulz, Simon Gog

Übungen:

Michael Axtmann

Institut für Theoretische Informatik, Algorithmik II

Web:

http://algo2.itl.kit.edu/AlgorithmenII_WS16.php
1 Algorithm Engineering

A detailed definition

- in general
  [with Kurt Mehlhorn, Rolf Möhring, Petra Mutzel, Dorothea Wagner]

- A few examples, usually sorting

- A little bit on experimental methodology
(Caricatured) Traditional View: Algorithm Theory

Theory

models

design

analysis

perf. guarantees

deduction

Practice

implementation

applications
## Gaps Between Theory & Practice

<table>
<thead>
<tr>
<th>Theory</th>
<th>→→ →</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple</td>
<td></td>
<td>complex</td>
</tr>
<tr>
<td>simple</td>
<td></td>
<td>real</td>
</tr>
<tr>
<td>complex</td>
<td></td>
<td>simple</td>
</tr>
<tr>
<td>advanced</td>
<td></td>
<td>arrays,…</td>
</tr>
<tr>
<td>worst case</td>
<td></td>
<td>inputs</td>
</tr>
<tr>
<td>asympt.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Theory
- Simple
- Complex
- Advanced
- Worst case

### Practice
- Complex
- Real
- Simple
- Arrays,…
- Inputs

### Theory vs. Practice
- Application model vs. Machine model
- Algorithms vs. Data structures
- Complexity measure vs. Efficiency
- Asymptotic vs. Constant factors

**Notes:**
- Simple vs. Complex
- Machine model vs. Real
- Algorithms vs. Simple data structures
- For efficiency, max inputs, 42% constant factors.
Algorithmics as Algorithm Engineering

algorithm engineering

models

design

analysis

deduction

perf.- guarantees

experiments
Algorithmics as Algorithm Engineering

algorithm engineering

models

design

falsifiable hypotheses

induction

analysis

deduction

experiments

implementation

perf.-guarantees
Algorithmics as Algorithm Engineering

**Algorithm engineering**

- **realistic models**
  - **real. design**
    - falsifiable hypotheses
      - induction
      - experiments
    - implementation
  - deduction
    - perf. guarantees
  - analysis

**Algorithmics**
Algorithmics as Algorithm Engineering

algorithm engineering

realistic models

real Inputs

design

falsifiable hypotheses

induction

experiments

implementation

algorithm–libraries

real Inputs

analysis

deduction

perf.–guarantees
Algorithmics as Algorithm Engineering

algorithm engineering

- realistic models
  - design
  - experiments
    - real Inputs
      - applications
        - algorithm engineering
          - analysis
            - deduction
              - perf.- guarantees
                - algorithm libraries
                  - induction
                    - falsifiable hypotheses
                      - implementation
                        - appl. engin.
Bits of History

1843– Algorithms in theory and practice

1950s, 1960s Still infancy

1970s, 1980s Paper and pencil algorithm theory.
   Exceptions exist, e.g., [D. Johnson], [J. Bentley]

1986 Term used by [T. Beth],
   lecture “Algorithmentechnik” in Karlsruhe.

1988– Library of Efficient Data Types and Algorithms
   (LEDA) [K. Mehlhorn]

1997– Workshop on Algorithm Engineering
   ⇝ ESA applied track [G. Italiano]

1997 Term used in US policy paper [Aho, Johnson, Karp, et. al]

1998 Alex workshop in Italy ⇝ ALENEX
# Realistic Models

<table>
<thead>
<tr>
<th>Theory</th>
<th></th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple</td>
<td></td>
<td>appl. model</td>
</tr>
<tr>
<td>simple</td>
<td></td>
<td>machine model</td>
</tr>
</tbody>
</table>

- Careful refinements
- Try to preserve (partial) analyzability / simple results
Design

of algorithms that work well in practice

- simplicity
- reuse
- constant factors
- exploit easy instances
Analysis

- Constant factors matter
  Beispiel: quicksort

- Beyond worst case analysis

- Practical algorithms might be difficult to analyze
  (randomization, meta heuristics, ... )
Implementation

sanity check for algorithms!

Challenges

Semantic gaps:
- Abstract algorithm
- $\leftrightarrow$
- C++...
- $\leftrightarrow$
- hardware
Experiments

☐ sometimes a good *surrogate for analysis*

☐ *too much* rather than *too little* *output data*

☐ *reproducibility* (10 years!)

☐ *software engineering*

Stay tuned.
Algorithm Libraries — Challenges

- software engineering
- standardization, e.g. java.util, C++ STL and BOOST
- performance ↔ generality ↔ simplicity
- applications are a priori unknown
- result checking, verification
Problem Instances

Benchmark instances for **NP-hard** problems

- TSP
- Steiner-Tree
- SAT
- set covering
- graph partitioning
- ...

have proved essential for development of practical algorithms

**Strange:** much less real world instances for **polynomial problems**

(MST, shortest path, max flow, matching...)
Example: Sorting Benchmark (Indy)

100 byte records, 10 byte random keys, with file I/O

<table>
<thead>
<tr>
<th>Category</th>
<th>data volume</th>
<th>performance</th>
<th>improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraySort</td>
<td>100 000 GB</td>
<td>564 GB / min</td>
<td>17×</td>
</tr>
<tr>
<td>MinuteSort</td>
<td>955 GB</td>
<td>955 GB / min</td>
<td>&gt; 10×</td>
</tr>
<tr>
<td>JouleSort</td>
<td>100 000 GB</td>
<td>3 400 Recs/Joule</td>
<td>??×</td>
</tr>
<tr>
<td>JouleSort</td>
<td>1 000 GB</td>
<td>17 500 Recs/Joule</td>
<td>5.1×</td>
</tr>
<tr>
<td>JouleSort</td>
<td>100 GB</td>
<td>39 800 Recs/Joule</td>
<td>3.4×</td>
</tr>
<tr>
<td>JouleSort</td>
<td>10 GB</td>
<td>43 500 Recs/Joule</td>
<td>5.7×</td>
</tr>
</tbody>
</table>

Also: PennySort
GraySort: inplace multiway mergesort, exact splitting

Xeon Xeon 16 GB RAM

240 GB

Infiniband switch

400 MB / s node all–all
JouleSort

- Intel Atom N330
- 4 GB RAM
- 4 × 256 GB SSD (SuperTalent)

Algorithm similar to GraySort
Applications that “Change the World”

Algorithmics has the potential to SHAPE applications
(not just the other way round)

Bioinformatics: sequencing, proteomics, phylogenetic trees,...

Information Retrieval: Searching, ranking,...

Traffic Planning: navigation, flow optimization,
adaptive toll, disruption management

Geographic Information Systems: agriculture, environmental protection,
disaster management, tourism,...

Communication Networks: mobile, P2P, cloud, selfish users,...
Conclusion:

Algorithm Engineering ↔ Algorithm Theory

- **algorithm engineering** is a wider view on **algorithmics**
  (but no revolution. None of the ingredients is really new)

- **rich methodology**

- better coupling to **applications**

- **experimental algorithmics** ≪ algorithm engineering

- **algorithm theory** ⊂ algorithm engineering

- sometimes **different theoretical questions**

- algorithm theory may still yield the strongest, deepest and most persistent results within algorithm engineering
More On Experimental Methodology

Scientific Method:

- Experiment need a possible outcome that falsifies a hypothesis

- Reproducible
  - keep data/code for at least 10 years
  - documentation (aka laboratory journal (Laborbuch))
  - clear and detailed description in papers / TRs
  - share instances and code
Quality Criteria

☐ Beat the state of the art, globally – (not your own toy codes or the toy codes used in your community!)

☐ Clearly demonstrate this!

- both codes use same data ideally from accepted benchmarks (not just your favorite data!)
- comparable machines or fair (conservative) scaling
- Avoid incomparabilities like: “Yeah we have worse quality but are twice as fast”
- real world data wherever possible
- as much different inputs as possible
- its fine if you are better just on some (important) inputs
Not Here but Important

- describing the setup
- finding sources of measurement errors
- reducing measurement errors (averaging, median, unloaded machine...)
- measurements in the creative phase of experimental algorithmics.
The Starting Point

- (Several) Algorithm(s)

- A few quantities to be measured: time, space, solution quality, comparisons, cache faults, ... There may also be measurement errors.

- An unlimited number of potential inputs. $\rightsquigarrow$ condense to a few characteristic ones (size, $|V|$, $|E|$, ... or problem instances from applications)

Usually there is not a lack but an abundance of data $\neq$ many other sciences
The Process

Waterfall model?

1. Design
2. Measurement
3. Interpretation

Perhaps the paper should at least look like that.
The Process

- Eventually stop asking questions (Advisors/Referees listen !)
- build measurement tools
- automate (re)measurements
- Choice of Experiments driven by risk and opportunity
- Distinguish mode
  - explorative: many different parameter settings, interactive, short turnaround times
  - consolidating: many large instances, standardized measurement conditions, batch mode, many machines
Of Risks and Opportunities

Example: Hypothesis $\Rightarrow$ my algorithm is the best

big risk: untried main competitor

small risk: tuning of a subroutine that takes 20 % of the time.

big opportunity: use algorithm for a new application

$\Rightarrow$ new input instances