Experimental Approaches in Computer Science

Dror Feitelson
Hebrew University

Lecture 11 – Experimental Algorithmics

Traditional approach:

"Quicksort is $O(n^2)$"

- What this means:

  Quicksort will perform at most cn^2 operations when sorting n numbers, regardless of programming language, programmer experience, or the specific input sequence

- This is a universal truth
- Note, however, that
  - c is not specified
  - This is the worst case, and might not be representative of common cases

Misleading complexity analysis – only huge *n*:

Minimum spanning tree

- Prim's algorithm has complexity O(E lg V)

  [Start with any node, and iteratively add lowest cost edge to a node that is not in the tree yet. Complexity depends on appropriate data structure]

- Fredman and Tarjan have an algorithm with complexity O(E lg* V)

  [Very complicated]

- In practice, the improvement is only seen for dense graphs with more than 1000000 nodes

  [Moret & Shapiro, DIMACS 1994]

Misleading complexity analysis – drowned by constant factor:

Test if one graph is a minor of another

- Robertson & Seymour give a cubic algorithm
- However, it has a constant of 10^{150}

Sorting network

- Ajtai, Komlos, and Szemeredi show an optimal O(log *n*)-depth construction
- Based on expander graphs
- Huge constants make it impractical

Misleading complexity analysis – worst case is uncommon

- Linear programming
  - The simplex method has an exponential worst case running time
  - However, it has a low running time for practically all naturally occurring inputs
- Similar situations exist for many NP-complete problems
  - Approximations may be available for most inputs
  - The problematic inputs may be rare and uninteresting

Missing complexity analysis does not necessarily imply bad performance

Minimize edge crossings when drawing bipartite graphs

- Problem is NP-complete
- Algorithm with no known *constant* approximation ratio leads to better results than algorithm with proven low constant approximation ratio

[Demetrescu & Finocchi, ALENEX 2000]

Misleading complexity analysis – does not take mundane implementation issues into account

Locality and cache effects

- May have dramatic effect on performance
- However, can be very hard to predict and analyze
- Partly due to complex parameterization of cache structures (set sizes, associativity, multiple levels)

Naive optimizations – implicit assumptions may be wrong

Code structure vs. cache effects

- Optimizations often geared to reduce instruction counts so as to accelerate execution
- This may lead to smaller code blocks and using less data in each basic block
- May result is reduced cache locality and subsequent longer running time

The experimental approach

- Emphasize real-world results, including constants
  - How much time will it really run?
- Emphasize common case rather than worst case
  - Also, how common is the common case?
  - Show distribution rather than just one data point
- Price is possible dependence on platform being used
  - Might restrict applicability
  - Typically good enough for qualitative comparisons

Basic methodology: Use a good implementation

- When studying an algorithm, make it relevant
  - Finding that an inefficient implementation is bad is not interesting
  - Studying a bad implementation is misleading and confounds the issues
- When comparing options, make it fair
  - Want to compare algorithms, not implementations
  - Need to invest similar amounts of effort

Basic methodology: Use representative input instances

- Behavior on random inputs may differ from behavior on real ones
  - Real world inputs may tend to be more structured
  - Such structure may provide opportunity for special optimizations
  - Or such structure may be harder to handle
- Similar to need for representative workloads

Basic methodology: Perform good measurements

- Use repetitions and calculate confidence intervals
  - Repetitions are over multiple representative inputs
  - Need to note whether distribution is bell-shaped or has a tail
- Remove outliers?
  - Not if they are important real cases
  - Yes if they reflect interference with measurement

Basic methodology: Report full details

- Allow for reproducibility
  - If others can reproduce it this increases our confidence
  - Not the same as replication, where others simply run your code on your inputs
- Include platform details, implementation details