Empirical Analysis

IE 496 Lecture 7

Reading for This Lecture

- Paper by Dolan and Moré
- Paper by Hooker

Empirical Analysis of Algorithms

- In practice, we will often need to resort to empirical rather than theoretical analysis to compare algorithms.
  - We may want to know something about performance of the algorithm “on average” for real instances.
  - Our model of computation may not capture important effects of the hardware architecture that arise in practice.
  - There may be implementational details that affect constant factors and are not captured by asymptotic analysis.
- For this purpose, we need a methodology for comparing algorithms based on real-world performance.

Issues to Consider

- Empirical analysis introduces many more factors that need to be controlled for in some way.
  - Test platform (hardware, language, compiler)
  - Measures of performance (what to compare)
  - Benchmark test set (what instances to test on)
  - Algorithmic parameters
  - Implementational details
- It is much less obvious how to perform a rigorous analysis in the presence of so many factors.
- Practical considerations prevent complete testing.

Measures of Performance

- For the time being, we focus on sequential algorithms.
- What is an appropriate measure of performance?
- What is the goal?
  - Compare two algorithms.
  - Improve the implementation of a single algorithm.
- Possible measures
  - Empirical running time (CPU time, wallclock)
  - Representative operation counts

Measuring Time

- There are three relevant measures of time taken by a process.
  - *User time* measures the amount of time (number of cycles taken by a process in “user mode.”
  - *System time* the time taken by the kernel executing on behalf of the process.
  - *Wallclock time* is the total “real” time taken to execute the process.
- Generally speaking, user time is the most relevant, though it ignores some important operations (I/O, etc.).
- Wallclock time should be used cautiously/sparingly, but may be necessary for assessment of parallel codes,

Representative Operation Counts

- In some cases, we may want to count operations, rather than time
  - Identify bottlenecks
  - Counterpart to theoretical analysis
- What operations should we count?
  - Profilers can count function calls and executions of individual lines of code to identify bottlenecks.
  - We may know a priori what operations we want to measure (example: comparisons and swaps in sorting).

Test Sets

- It is crucial to choose your test set well.
- The instances must be chosen carefully in order to allow proper conclusions to be drawn.
- We must pay close attention to their size, inherent difficulty, and other important structural properties.
- This is especially important if we are trying to distinguish among multiple algorithms.
- Example: Sorting

Comparing Algorithms

- Given a performance measure and a test set, the question still arises how to decide which algorithm is “better.”
- We can do the comparison using some sort of summary statistic.
  - Arithmetic mean
  - Geometric mean
  - Variance
- These statistics hide information useful for comparison.

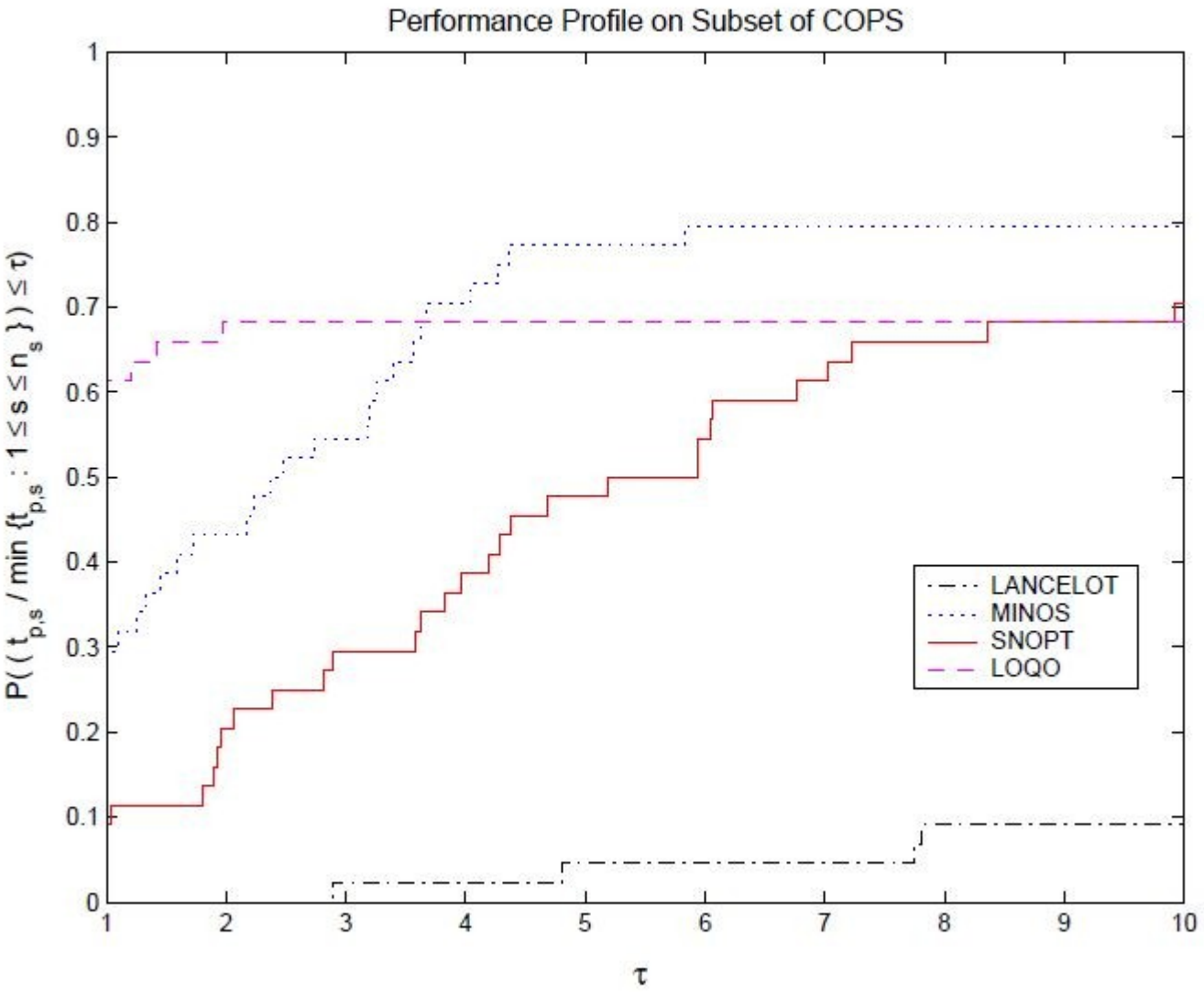
Accounting for Stochasticity

- In empirical analysis, we must take account of the fact that running times are inherently stochastic.
- If we are measuring wallclock time, this may vary substantially for seemingly identical executions.
- In the case of parallel processing, stochasticity may also arise due to asynchronism (order of operations).
- In such case, multiple identical runs may be used to estimate the affect of this randomness.
- If necessary, statistical analysis may be used to analyze the results, but this is beyond the scope of this course.

Performance Profiles

- Performance profiles allow comparison of algorithms across an entire test set without loss of information.
- They provide a visual summary of how algorithms compare on a performance measure across a test set.

Example Performance Profile



Empirical versus Theoretical Analysis

- For sequential algorithms, asymptotic analysis is often good enough for choosing between algorithms.
- It is less ideal with respect to tuning of implementational details.
- For parallel algorithms, asymptotic analysis is far more problematic.
- The details not captured by the model of computation can matter much more.
- There is an additional dimension on which we must compare algorithms: *scalability*.