A Theoretician's Guide to the Experimental Analysis of Algorithms

David S. Johnson

AT&T Labs - Research

http://www.research.att.com/dsj/

November 25, 2001

Synopsis

- This paper presents an informal discussion of issues that arise when one attempts to analyze algorithms experimentally.

- It is intended to be of use to researchers from all fields who want to study algorithms experimentally.

- It has two goals:
  - First, to provide a useful guide to new experimentalists about how such work can best be performed and written up,
  - Second, to challenge current researchers to think about whether their own work might be improved from a scientific point of view.

Introduction

- Three different approaches for analyzing algorithms:
  - Worst-case analysis,
  - Average-case analysis, and
  - Experimental analysis

- Experimental analysis treated almost as an afterthought.

- Until recently, experimental analysis of algorithms has been almost invisible in the theoretical computer science literature, although it <u>dominates algorithmic research</u> in most other areas of computer science and related fields such as operations research.

Theory to Practice

- The benefits of theoretical modes of analysis i.e. through the mathematical approach have been many, i.e.
  - Not only In added understanding of old algorithms but in the invention of new algorithms and data structures,
  - Ones that have served us well as faster machines and larger memories have allowed us to attack larger problem instances.

- However, recently there has been an upswing in interest in experimental work in the theoretical computer science community because of:
  - A growing recognition that theoretical results cannot tell the full story about real-world algorithmic performance.
  - Encouragement for experimentation has come both from researchers and from funding agencies who view experimentation as providing a pathway from theory into practice.

Implementation of an Algorithm

- The field of experimental analysis is fraught with pitfalls. In many ways, the implementation of an algorithm is the easy part.

- The hard part is successfully using that implementation to produce meaningful and valuable (and publishable!) research results.

- Four basic reasons to implement an algorithm

Implementation of an Algorithm

1. To use the code in a particular application (This includes experimental mathematics, where the “application" is the generation and proof of mathematical conjectures and we are primarily interested in the output of the algorithm, not its efficiency.)

2. To provide evidence of the superiority of your algorithmic ideas. Here, results are published for standard benchmark instances to illustrate the desirability of this algorithm in comparison to previous competitors.

3. To better understand the strengths, weaknesses, and operation of interesting algorithmic ideas in practice.

4. To generate conjectures about the average-case behavior of algorithms under specific instance distributions where direct probabilistic analysis is too hard.

Ten General Principles

Ten basic (and overlapping) principles that should govern the writing of experimental papers:

- 1. Perform newsworthy experiments.
- 2. Tie your paper to the literature.
- 3. Use instance test beds that can support general conclusions.
- 4. Use efficient and effective experimental designs.
- 5. Use reasonably efficient implementations.
- 6. Ensure reproducibility.
- 7. Ensure comparability.
- 8. Report the full story.
- 9. Draw well-justifed conclusions and look for explanations.
- 10. Present your data in informative ways.

Principle-1: Perform Newsworthy Experiments

- This principle applies to all scientific papers: the results should be new and of interest and/or use to some reasonably-sized collection of readers.
  - For instance, it is much harder to justify experimental work on problems with no direct applications than it is to justify theoretical work on such problems.

- A second dimension of newsworthiness has to do with the algorithms studied. A key question about any algorithm is whether it might be a useful alternative in practice.

- A third dimension of newsworthiness has to do with the generality, relevance, and credibility of the results obtained and the conclusions derivable from them.

Principle-1: Perform Newsworthy Experiments

- Practices leading to **Waste of Time**:
  - Dealing with Dominated Algorithms
  - Devoting too much computation to the wrong questions
  - Getting into an endless loop in your experimentation
  - Start by using randomly generated instances to evaluate the behavior of algorithms, but end up using algorithms to investigate the properties of randomly generated instances.

- Practices leading to **Misguidance**:
  - Authors and Referees who don’t do their homework

- **Suggestions**:
  - Think before you compute
  - Use exploratory experimentation to find good questions

Principle-2: Tie Your Paper to the Literature

- One key component in establishing the newsworthiness of a paper is placing it in proper context with respect to the literature on the problem being studied.

- Indeed, before undertaking any experimental project, you should do your best to discover and thoroughly study the prior literature if it exists.

- This should go without saying, but surprisingly many papers have failed because of lack of diligence on this point.

Principle-3: Use Instance Test beds that can Support General Conclusions

- There are basically two types of test instances available to experimenters: particular instances from real-world applications (or pseudo-applications) and randomly generated instances.
- The former are typically found in standard repositories, but might come from private sources proprietary to the author.
- The latter are provided by instance generators that, given a seed and a list of instance parameters, such as size and perhaps other details of instance structure, produce a random instance consistent with those parameters.

Principle-3: Use Instance Test beds that can Support General Conclusions

- Practices leading to **Misguidance**:
  - Concentration on unstructured random instances
  - The millisecond test bed
  - The already-solved test bed

Principle-4: Use Efficient and Effective Experimental Designs

- There exist important techniques that will not only help you reduce the amount of computation you need to perform but also allow you to get broader and more accurate answers.

Principle-4: Use Efficient and Effective Experimental Designs

- **Suggestions:**
  - Use variance reduction technique
  - Use bootstrapping to evaluate multiple-run heuristics
  - Use self-documenting programs

Principle-5: Use Reasonably Efficient Implementations

- Although it would at first glance seem obvious that we should want to use efficient implementations (especially in a field like algorithm design where efficiency is one of our main goals), efficiency does come at a cost in programming effort and there are several situations in which researchers have argued that they should be allowed to settle for less.

Principle-5: Use Reasonably Efficient Implementations

- Practices leading to **Waste of Time**:
  - Too much code tuning

- Practices leading to **Misguidance**:
  - Claiming inadequate programming time/ability as an excuse

Principle-6: Ensure Reproducibility

- As with all scientific studies, a key part of an experimental paper is the reproducibility of the results.

- But what does "reproducibility" mean in the context of the experimental analysis of algorithms?

- In the strictest sense it means that if you ran the same code on the same instances on the same machine/compiler/operating system/system load combination you would get the same running time, operation counts, solution quality (or the same averages, in the case of a randomized algorithm).

Principle-6: Ensure Reproducibility

- Practices leading to **Misguidance**:
    - Supplied code that doesn’t match a paper’s description of it
    - Irreproducible standards of comparison
    - Using running time as a stopping criterion
    - Using the optimal solution value as a stopping criterion
    - Hand-tuned algorithm parameters
    - The one-run study
    - Using the best result found as an evaluation criterion

Principle-7: Ensure Comparability

- This Principle is essentially the reverse side of the Principle about tying your paper to the literature.
- That earlier principle referred to the past literature. This one refers to the future.
- You should write your papers (and to the extent possible make relevant data, code, and instances publicly available in a long-term manner) so that future researchers (yourself included) can accurately compare their results for new algorithms/instances to your results.

Principle-7: Ensure Comparability

- Practices leading to **Waste of Time**:
  - Lost Code/data

- Practices leading to **Misguidance**:
  - The un-calibrated machine
  - The lost test bed

- **Suggestions**:
  - Use benchmark codes to calibrate machine speeds

Principle-8: Report the Full Story

- Although it is typically overkill to present all your raw data, if you are overly selective you may fail to reach accurate conclusions or adequately support them.
- You may also impair your paper's reproducibility and comparability.
- If you report only averages, you should say how many instances (and how many runs in the case of randomized algorithms) you based the averages on. If the precise values of these averages are important for your conclusions, you should also provide information about the distribution of the results.
- The latter might simply be standard deviations if the data appears roughly normally distributed, but might need to be more pictorial (histograms, boxplots) if the distribution is more idiosyncratic.
- And any scaling or normalizing of measurements should be carefully explained so that the raw averages can be regenerated if desired.
- In particular, although pictorial summaries of data can often be the best way to support your overall conclusions, this should not typically be the only presentation of the data since pictures typically lack precision.
- Tables of data values/averages may be relegated to an appendix, but should be included.

Principle-8: Report the Full Story

- Practices leading to **Misguidance**:
  - False precision
  - Unremarked anomalies
  - The ex post facto stopping criterion
  - Failure to report overall running times

Principle-9: Draw Well-Justified Conclusions and Look for Explanations

- The purpose of doing experiments with algorithms is presumably to learn something about their performance, and so a good experimental paper will have conclusions to state and support.

- It is not enough simply to perform tests, tabulate your results, and leave the reader the job of finding the implications of the data. (If there aren't any implications that you can find or express, then you've done the wrong experiments and shouldn't be writing a paper yet, as you've failed the "newsworthiness" test.)

- At a minimum, one should be able to summarize patterns in the data. Ideally, one might also be able to pose more general conjectures that the data supports and which subsequent research (experimental or theoretical) may try to confirm or deny.

Principle-9: Draw Well-Justified Conclusions and Look for Explanations

- Practices leading to **Misguidance**:
  - Data without interpretation
  - Conclusions without support
  - Myopic approaches to asymptopia
- **Suggestions**:
  - Use profiling to understand running times

Principle-10: Present Your Data in Informative Ways

- The best way to support your conclusions (and sometimes also the easiest way to derive them) is to display your data in such a way as to highlight the trends it exhibits, the distinctions it makes, and so forth.

- There are many good display techniques depending on the types of points one wants to make.

Principle-10: Present Your Data in Informative Ways

- Practices leading to **Misguidance**:
  - Tables without pictures
  - Pictures without tables
  - Pictures yielding too little insight
  - Inadequately or confusingly labeled pictures
  - Pictures with too much information
  - Confusing pictorial metaphors
  - The spurious trend line
  - Poorly structured tables
  - Making your readers do the arithmetic
  - The undefined metric
  - Comparing apples with oranges
  - Detailed statistics on unimportant questions
  - Comparing approximation algorithms as to how often they find optima
  - Too much data

- **Suggestions**:
  - Display normalized running times