How to Present a Paper on Experimental Work with Algorithms

Catherine C. McGeoch* Bernard M.E. Moret†

September 2, 1999

Abstract

Inspired by Ian Parberry's "How to present a paper in theoretical computer science," (*SIGACT News* **19**, 2 (1988), pp. 42–47), we provide some advice on how to present results from experimental and empirical research on algorithms.

1 Introduction

This note is written primarily for researchers in algorithms who find themselves called upon to present the results of computational experiments. While there has been much recent growth in the amount and quality of experimental research on algorithms, there is still some uncertainty about how to describe the research and present the conclusions.

For general advice on presenting papers in theoretical computer science, read Ian Parberry's excellent paper [6]. Here we focus on aspects directly relevant to experimentation and data analysis.

Of course, the quality of the talk depends on the quality of the research. For advice on conducting respectable experimental research on algorithms, read McGeoch [5], or Barr *et al.* [1], or the articles on methodology to appear in [4] and available on-line.

2 What to Say

State the problem clearly and give your rationale for using an experimental approach. A crucial part of the description of the problem is your motivation for conducting experiments. Presumably, the problem does not lend itself to a precise analytical characterization—why? Is it due to properties of the data (as in an application)? of the platform (as in a study of caching effects or consistency across platforms)? or to the very complex nature of your algorithm (as is common in optimization heuristics)? Not every problem is best approached experimentally, so justify your choice.

Provide context for the research—has the algorithm been studied previously? Has it been implemented and tested? What have others done? If your problem is entirely new, what are you trying to achieve through experimentation? What have you done to enable you to go beyond "it runs fairly fast and returns pretty good solutions"?

*Department of Mathematics and Computer Science, Amherst College, Amherst, MA 01002, ccm@cs.amherst.edu

†Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, moret@cs.unm.edu

If you have been working on an application, describe the input data. What makes it different from what could be randomly generated? How has it been collected? What steps have you taken to ensure good sampling?

Describe and motivate the specifics of the experiments. The audience should now have a clear idea of why you are doing experiments; you must also explain what specific experiments you have conducted and why. Describe the general experimental environment; give the range of problem sizes and the names and ranges of other parameters; indicate and justify the number of random trials per test. The audience should be able to see the relevance of the experimental design to the questions you are investigating. Why these input classes? Why did you measure this quantity and not that one? Are these problem sizes typical in practice? Why did you use only five trials per data point?

Mention enough details of the experiments. If runtime statistics are important, mention the machine, environment, compiler version and optimization level, maybe even memory sizes. Even if you do not report running times, you should say how much total CPU time went into the experiments and approximately how long the largest problems took to run. If your experiments are specifically targeted at characterizing platform effects, go into more detail about the attributes of these platforms and how your measurements might capture some of these effects; discuss possible artifacts caused by your instrumentation.

If the experiments are being used to make analytical conjectures, mention what steps you took to avoid machine dependence in the results. Address any fears that the random generator, machine precision, caching effects, or other environmental problems (particularly if you ran your code on a shared machine) might have skewed your results. Describe the validation experiments you performed with an alternative backup implementation.

As a service to future experimenters mention any difficulties or unusual aspects of the experiment, and what you did about them.

But do not mention too many details! Do not slap code up on the slide; if something in the code is crucial, show the relevant snippet. Do not pile on environment and machine characteristics unless they are directly relevant. Do not list every parameter setting you tested. Do not present every last data value you gathered—instead, select for presentation the data that most clearly illustrates your claims; but make a full disclosure of any discordance and attempt to explain it.

In most cases you will want to discuss briefly the algorithms you developed or implemented (with proper attribution). Please realize that the emphasis should be on the experimental work: there is not enough time both to describe the algorithms at the level of detail usually seen in theory talks and to discuss adequately the experiments and results, so cut back on the algorithmic description.

The appropriate level of algorithmic detail depends on the focus of the research. If you are presenting a new algorithm, spend a little more time on it—the experiments will probably be exploratory. If you conducted a comparison of several algorithms, give a two-sentence explanation of the properties of each; the data will be complex and rich, so spend a lot of time on the tests and their results. If you worked on an application, spend more time discussing the nature of the application as it affects the structure of the input data.

Draw conclusions and support them. You cannot just present a graph or table of numbers and call it "the results." (In fact, you cannot present a table of numbers at all—see below.) It is your responsibility to draw conclusions, identify relationships, and describe patterns that you have discovered; graphs should be used to support your observations. The supporting presentation should therefore be directly relevant to the observation—do not make your audience do mental gyrations to compare the data to your claim. If you say A is generally 7 times better than B, do not show a graph with 20 scores for each and expect the audience to do the arithmetic; instead present the ratios to show the factor directly. Match the scale of the illustration to the assertion: if "Algorithm A is about twice as fast as algorithm B on these inputs," then the supporting picture should be scaled to show "about twice."

The point is not to overwhelm the audience with numbing lists of values and details, but to highlight your main findings or contributions. You can take advantage of two projectors here by showing the conclusions on one and various supporting data on the other. Also think about creative ways to use slides, with overlays and such; with a computer projection system, you can use multiple windows, text overlays, moving highlights, and semi-transparent graphs for the same purpose.

But make sure that the support is real! Do not overgeneralize. (For instance, do not extend your observations on planar graphs to claims about other kinds of graph families; do not extend your runtime observations to other computational environments.) Be particularly cautious about extending your results to larger problem sizes—the literature is full of asymptotic conjectures based on experimental results that were later invalidated by other experimental results or by analysis.

Whenever possible, use statistical analysis to estimate the significance of your results. In addition to averages, present information about level of significance, confidence intervals, variance, outliers, and extreme points, as appropriate.

A graph is worth a thousand table entries. It is a matter of experimental integrity to present, as much as possible, all of your data, not just summary statistics. Of course, it is rarely possible to show every single datum, but make every effort to show the whole picture, including variation from the mean, outliers, and data points that do not support your claims. Graphs, especially scatter plots, make it possible for the audience to assimilate the mass of information you collected.

We hereby ban the use of large tables of numbers from all presentations of computational experiments on algorithms (yes, even in proceedings). If you have more than four data points to present, use a graph. (If you really have only four data points, in which case a graph may be misleading, you should question whether you have enough experimental evidence to draw any kind of conclusions.) Tables are not readable in an oral presentation and are very hard to interpret even in print. Graphs provide the best way for the audience to see trends, ratios, and patterns in data. If someone in the audience simply has to have the actual numbers, you can make them available on the web.

Scatter plots are extremely helpful in understanding the distribution of the data and showing outliers: plot the data against the predicted or extrapolated behavior; or normalize it and allow the audience to evaluate deviations from the horizontal; or superimpose the prediction on the observations.

With additional time investment, you can use a laptop and projection system to show an animation of the data. While animations acre often the best means for conveying your results, the lack of proper tools for building animations currently makes this approach too time-consuming, so you may want to resort to multiple snapshots in a succession of slides.

Explain what are we seeing. What do those points represent—is that solution quality or runtime? Seconds or microseconds? Is it a ratio or a difference? For each graph, state the quantities depicted and their units and scale (linear, logarithmic, etc.) on abscissa and ordinate. Keep units and values simple, through normalization if necessary. If you are showing means, indicate the number of trials as part of the legend and try to include variance brackets or some indication of the deviation from the mean.

When you are presenting comparisons of several algorithms or implementation strategies, clearly tie a name to each strategy and clearly label each on the graph (color can be very useful here). For a complicated or sophisticated graphic you may want to start with a simple example of how the graphic works, and then move to the real data.

3 Organizing the Talk

Overview. Introduce the problem, the data, the algorithms. State the open questions, which ones are addressed in this talk, and your general conclusions.

Introduction. Present more about each algorithm or implementation strategy, so the audience can distinguish which is which and understand what is known already from the literature; but definitely refrain from giving a detailed description of the algorithms.

The setup. Describe the experiments in general. What questions are you addressing? (Write down the questions on a slide.) How much total time did the experiments take? Clearly show which algorithm or property tested goes by which name. Give an overview of the environment or any aspects of the research that are held constant throughout the individual experiments.

The experiments.

- Introduce each experiment. What question is being investigated?
- What parameters are you using and what values are you choosing for them? What data are used and how were they collected or generated?
- How do measured values relate to analytical values?
- Present the conclusions, observations, claims, as declarative sentences written down. Show relevant supporting data in a clean graphical representation.
- Point out surprises or anomalies. Address any unusual data that does not support your conclusion or stands in contrast to the general trends—foresee possible substantive questions and address them first!
- What are the limitations this experiment? Under what circumstances would your results be significant? irrelevant?

Conclusions. Summarize your main observations. Put them in the context of previous research (showing what new information have you learned) and in the context of future research (suggesting what should be done next). Describe what is missing, what is still unresolved, and how the experiments could be extended.

Variations on the theme.

Algorithm engineering: If you fine-tuned an existing algorithm for a library or for utmost performance, you may want to give a very sequential presentation: address each modification you tried and its results, leading to the next modification.

Application: Distinguish between a "real" application (in which your implementation actually gets used) from a "model" application (in which simulated data and a simplified model are used). In a model application, spend a fair amount of time discussing data characterization and generation.

Characterizing platform effects: You could be doing this in order to develop robust library modules—in which case you need to state your specific goals and trade-offs (e.g., how much slower your module is than a tailored, one-shot implementation)—or in order to develop a model for the analysis and design of platform-aware (for instance, cache-aware) algorithms, in which case you need to state what aspects of the platforms you are considering, how common these parameters are, and so forth.

Algorithm design: If you are presenting a more traditional "new algorithm" or "improved analysis" talk (with just some preliminary experimentation to establish feasibility, suggest the right ballpark for parameter values, or indicate extensions to the analysis), make this clear right away. You need not present experimental details or specific results—simply weave your findings in the description of the algorithm design or analysis, emphasizing the interaction between your experiments and your theoretical developments.

Choosing the best algorithm or implementation: Since few algorithms are truly general-purpose, be sure to define your intended coverage (sparse graphs, or disjoint objects, or small alphabets, etc.) and to discuss how you achieved this coverage with your test data.

Human experiments: If you conducted experiments with human subjects (e.g., to identify which optimization criteria lead to the most understandable graph drawings or to estimate the contribution that a "man-in-the-loop" can make to an optimization process), spend a lot of time describing the methodology and assume that your audience is mostly unfamiliar with it. Be as quantitative as possible.

4 Nuts and Bolts

Review the article of Parberry [6].

Time management. As a general rule, plan to spend no more than half the time describing the problem, the open questions, and the experimental setup. Leave plenty of time for presenting the results, their implications, and your conclusions. Experimental work also tends to generate more questions from the audience than more traditional talks on algorithm design.

Write down the main points. People need to have a visual memory of the most important points. Help them by listing the major questions addressed in your research. Help them remember which algorithm is which by providing a list matching names to properties. Help them remember your important conclusions by writing them on a slide—do not rely on your commentary. One should be able to get the important points just by *looking* at your slides.

Learn about graphical presentation. Read a good book, such as Cleveland [3] or Chambers *et al.* [2]. Think about font size: do not show tiny labels, points that disappear in the graph, overlapping symbol blobs, or 50 lines of tiny code. Use color: even if you have a black-and-white printer, you can use a color marker to highlight some data points or interesting values. Ask what means of information display will be available (one or more overhead projectors? a computer projection system? closed-circuit TV?) and plan your presentation to take advantage of all available means.

Take advantage of technology. Most conferences now provide a computer projection system. With a laptop, you can use a large range of colors, multiple windows, animated overlays, flexible highlighting, or even full-fledged animations (including brief demos of a part of your work), etc. Keep in mind that screens are biased horizontally, unlike the traditional use of a page—use landscape mode in order to take advantage of the screen. Be mindful of limitations; for instance, hue and saturation are highly variable among projection systems, so use only easily distinguishable colors and ensure that backgrounds and foregrounds offer sufficient contrast.

References

[1] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart, "Designing and reporting on computational experiments with heuristic methods," *J. Heuristics* **1**, 1 (1995), pp. 9–32.

[2] J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey. *Graphical Methods for Data Analysis*. Duxbury Press, Boston (1983).

[3] W.S. Cleveland. *Elements of Graphing Data*. Wadsworth, Monterey, CA (1985).

[4] D.S. Johnson and C.C. McGeoch, eds. *Proc. 5th DIMACS Challenge*, AMS Society Press, to appear (2000). Draft methodology papers are available at the website `www.cs.amherst.edu/~dsj/methday.html`.

[5] C.C. McGeoch, "Toward an experimental method for algorithm simulation," *INFORMS J. Comput.* **1**, 1 (1996), pp. 1–15.

[6] I.M. Parberry, "How to present a paper in theoretical computer science: a speaker's guide for students," *SIGACT News* **19**, 2 (1988), pp. 42–47. Also available at `hercule.csci.unt.edu/ian/guides/speaker.html`.