Experimental Algorithmics for Undergraduates

Catherine C. McGeoch
Department of Mathematics and Computer Science
Amherst College, Amherst, MA
ccm@cs.amherst.edu

1. INTRODUCTION

As a general rule, the standard undergraduate curriculum in computer science gives short shrift to topics in experimental methodology. This is unfortunate because, besides the obvious need for more and better education on in this area, experimental projects are interesting and fun. My (admittedly quite localized) experience suggests that students who have success with small experimental research projects – we call it research if the professor doesn't know beforehand what the outcome will be – become more interested in taking additional computer science courses and in applying to graduate school.

In this paper I will focus on ideas for incorporating student work in experimental algorithmics into the undergraduate curriculum.

What is Experimental Algorithmics?

Research in experimental algorithmics lies somewhere between traditional *theoretical* and *empirical* approaches to algorithm analysis. Classically, theoretical analysis emphasizes the derivation of asymptotic upper bounds on a function relating input size n to the number of "dominant operations" performed by an algorithm. On the other hand, the empiricist is interested in measuring the real-world performance of an implemented algorithm – a program – on a given platform, for a given set of "typical case" inputs.

The experimental approach combines the theoretician's emphasis on discovering functions to describe the cause-and-effect relationship between input properties and algorithm performance, with the empiricist's interest in realistic results. The related term *algorithm engineering* refers to this experimental methodology when applied to the problem of algorithm design (rather than analysis).

The next section describes a course in experimental algorithmics that I have taught over the past few years. The following section presents ideas for embedding experimental algorithmics into standard courses in the undergraduate computer science curriculum.

2. A COURSE IN EXPERIMENTAL ALGORITHMICS

At Amherst we offer an upper level Seminar course with topic that varies from year to year. Most recently, this course has been primarily devoted to work in experimental computer science, including topics in systems as well as experimental algorithmics. This section describes the most recent course in Experimental Algorithmics, which was titled "Algorithms in the Real World."

Our junior-level Algorithms course was prerequesite for this seminar. Student work was carried out on a Linux platform. We used the SPlus statistical package (a freeware version called R is widely available), which provides outstanding graphical data analysis tools. The syllabus was divided into three equal-sized units: Algorithm Engineering; Methods, Statistics, and Data Analysis; and Readings on the State of the Art.

Algorithm Engineering. In this section, students tackle the question: What matters (most) to algorithm and program efficiency? We consider four primary sources of efficiency, organized according to the *algorithm design hierarchy*:

- **Systems**. This layer contains broad strategy decisions about project decomposition and system support.
- **Algorithms and Data Structures.** This layer includes classic asymptotic analysis, and also considers problem-specific issues such as appropriate choice of data structures and analyses of memory- and I/O- efficiency.
- **Code Tuning.** This layer concerns the application of standard code tuning techniques to trim constant factors from program runtimes.
- **Platforms.** This layer considers ways to speed up already-implemented by improving the environment in which it runs.

Readings for this section of the course include: A small collection of papers in the literature that report on and compare

speedups due to different strategies at different layers; and a set of articles illustrating the application of these techniques. The general conclusion drawn from these readings is that algorithmic choices have the most impact on performance, but that great speedups can be found

For a first project, students are asked to engineer an algorithm to solve the all-pairs shortest paths problem on graphical data from a particular application. (I recommend choosing a problem requiring at least a $\Theta(n^3)$ algorithmic solution so that good distinctions can be observed.) Students are shown several ways to measure algorithmic and program performance, and they must design experiments to evaluate their solutions. Each student writes a paper and makes a presentation to the class.

Methodology, Statistics, Data Analysis. One outcome of the first project is that students realize there are many pitfalls in algorithm experimentation. Their result are usually not generalizable or even comparable. This experience sets up the second phase of the course, which concerns experimental methodology, statistics, and data analysis. Readings in this course unit cover the following topics:

- Tools: Unix tools; the SPlus package; tour of resources on the web.
- Introductory statistics and basic experimental design.
- Advanced topics such as Variance Reduction Techniques, function-fitting, and residuals analysis.
- Methods: designing experiments, building experimentation platforms, special problems relating to NP-Hard problems.
- Data analysis, emphasizing graphical methods of analysis.

For their project in this section, each student must choose an algorithmic problem to investigate using experiments; carry out the experiment and report the results. Here are some example projects.

- Ana. Comparison of matrix decomposition strategies for Strassen's algorithm.
- Crystal. How does changing the neighborhood function affect TABU search in a graph coloring application?
- Dan. A comparison of data structures for a greedy algorithm for facilities location.
- Mark. How does choice of start state and step size affect convergence and outcome in an iterative algorithm for an application in physicis?
- Megan. Can LaMarca and Ladner's results for cache-sensitive sorting algorithms implemented in C be replicated in Java?

It is interesting to note that five of these six students decided to pursue graduate studies in computer science related areas. Two mentioned specifically that this small research experience was instrumental in their decision.

Readings on the State of the Art. The third section of the course asked students to read papers containing *examples* of experimental research on algorithms, and to critique those papers according to the techniques and methodologies they have learned. Not surprisingly, the students found, in general, that the state of the art in experimental analysis of algorithms is fairly primitive. Students found errors in statistical analysis, weaknesses in experimental designs, significant mis-matches between stated experimental goals and the actual experiments, and many other problems.

This section took place at the end of the course to give more time to work on their individual research projects. In addition to in-class discussion of research articles, each student wrote and presented a small literature-survey paper that compared two articles on a similar algorithmic topic.

3. EMBEDDING EXPERIMENTAL ALGORITHMICS IN STANDARD UNDERGRADUATE COURSES.

Experimental algorithmics is a rich source of topics for "senior honors" projects: it requires the student to read a small collection of research papers to get up to speed with the open questions; to design appropriate experiments; and to develop code to carry out the experiments. More importantly for undergraduates, the scope of the project can be scaled up or down according to the rate of student progress.

Here are some example thesis projects in experimental algorithmics that have been undertaken by Amherst Computer Science students in recent years.

- Crystal. Comparison of communication strategies when porting TABU Search to a distributed environment.
- Miro. An experimental comparison of dynamic graph connectivity algorithms.
- Ben. Investigation of strategies for speeding up an exhaustive-search algorithm proposed by a statistics professor.
- Andrew. Scheduling the NESCAC soccer season (a highly constrained problem) using genetic algorithms.
- Dan. Comparison of algorithms to solve a fault-tolerant extension of the facilities location problem.
- Tishya. Comparison of data structures for an algorithm to simulate fish schooling patterns.

In addition to the thesis project, individual faculty members incorporate small experimental projects (homework assignments and term papers) in introductory and required courses throughout the curriculum. Here are two example projects that I have used in low-level courses.

- **Introduction to programming I.** Students are given four already-written programs (binary search, two sorting algorithms, and Towers of Hanoi) and asked to estimate what input size would require 24 hours of running time.

  Their first attempts at measurement are ineffective and inconclusive. We then have a class discussion on good experimental design; students learn how to counters in code to produce less noisy data. Their second round of experiments produce much more satisfying results.

- **Data Structures.** Which binary tree balancing scheme works best for a particular search application? Students work in teams to develop their favorite tree-balancing ideas. Before measurement can begin, students are required to focus on the problem of *designing* useful and informative experiments: in addition to finding a good algorithmic solution, they must grapple with the problem of how they will know whether a good solution has been found.

While these example problems are not particularly innovative or unusual, it can be surprising to see how excited students can get about designing their own homework assignments. Class discussion about what properties should be measured, and which kinds of inputs constitute a fair test, can get quite lively when students are invested in the success of their own implementations.

4. FINAL REMARKS

Many would agree that undergraduate computer science students need much more exposure to experimental methods. One significant obstacle is the woeful lack of textbooks and guidelines for this type of material. But on the other hand, as this short article illustrates, the field of experimental algorithmics provides ample opportunities for developing large and small research projects in experimental algorithmics. Experience suggests that students react to these types of projects with enthusiasm and with increased interest in pursuing computer science.