Experiments as Research Validation – Have We Gone too Far?

Jeffrey D. Ullman, July 9, 2013

I recently submitted a paper to VLDB, and when I got the reviews back, I noticed that the review form now has a question referees are required to answer, about whether the experiments were well carried out, with choices like "believable" and "not believable." The reviewers had a bit of trouble with that question, because my paper had no experiments; it was a paper about computational complexity of MapReduce algorithms. Two of the reviewers said the nonexistent experiments were not believable, which is wrong – you have to see something to disbelieve it.

It appears the database community has now reached the point where experiments are no longer an option. There was a time when experiments were not normal in a computer-science research paper. There was even a term in the database community -- a "Wisconsin-style paper" -- for a paper in which a proposed algorithm was implemented and experiments regarding its performance were described. Kudos to the faculty at Wisconsin for seeing the value of using experimentation as a way to demonstrate the worth of certain ideas. However, now it appears the pendulum has swung way too far, to the point where experiments are considered the only way to validate ideas. It is time to restore the balance, where experiments are used when appropriate, and ideas that require analysis rather than experiments are handled appropriately, rather than "justified" by inappropriate and meaningless experiments.

Good ideas should stand on their own. Look at the two database ideas that have won the Turing award: the relational model and 2-phase locking (I know Jim Gray won for many contributions, but this one is, I think, the centerpiece). Neither paper was about experiments. Should we have rejected a paper that said "let's organize data into relations" because there was no experiment to prove that its queries were executable more efficiently? Would we want to reject the 2PL paper because it did not measure experimentally the space required by locking tables?

Please don't mistake this essay as saying "no paper should have experiments." The need for experiments in many situations is clear. Rather, let's consider what harm the overemphasis on experiments brings. First, experiments are conducted on particular data or under particular assumptions. They rarely tell you what happens in other situations. In contrast, when you do a formal analysis of the resources required by an algorithm, that analysis applies generally. In a well done analysis, you are forced to introduce the relevant parameters (amount of main memory, maximum number of

occurrences of a single value of a single attribute, e.g.) that characterize the true performance of the algorithm.

For example, long ago we discovered algorithms to sort in $O(n \log n)$ time. These algorithms were analyzed formally, using a realistic model of the main memory of a computer. We didn't have to run the algorithms and plot their running time to know they were better than the obvious $O(n^2)$ algorithms. There was a place for experimentation, and some investigations looked at matters such as how many elements must there be before Quicksort really beats Bubblesort. And of course when you ran out of main memory, the model no longer applied and you had an unpredicted increase in running time as you suddenly needed to move data to and from disk. Yet the basic $O(n \log n)$ idea still applies even when you use secondary storage.

But there is a more damaging effect of the seriousness with which we take experimental results. It encourages the writing of papers that really shouldn't be written, because they are so incremental and specialized that their use in practice is unlikely. There are many areas of database research where the nature of the data can vary greatly, and performance of different algorithms will vary with the data. Think of multidimensional indexes, or clustering, or even join algorithms. In research areas of this kind, it is very easy to find a special form of data and invent an algorithm that works well in this narrow special case. You then run your experiments on data for which your algorithm is best suited and compare it with others, which – surprise surprise – do not work as well. But were you to run your algorithm on the common cases, or random cases, you would do less well or not well at all. It doesn't matter; you can still publish yet another paper about yet another algorithm for doing this or that.

Suppose we expected, as an alternative or supplement to experiments, a formal analysis of the performance of the algorithm(s) described in a research paper. It would then be much harder to hide the fact that your algorithm works well only in some narrow special case. You would need to give expressions for the performance of the algorithm in all cases, and this expression would be compared with the analogous expressions for other algorithms. Because these expressions can't represent the details of the code that would implement the algorithms involved, you get only a big-oh estimate of the running times. But Big-oh analysis becomes progressively more relevant as the data size gets larger, and database research always focuses on the largest data anyway. Returning to the sorting example, if you know one algorithm is $O(n \log n)$ and another is $O(n^2)$, you might not know which is really better for small n, but you know for certain that the first is better for matters a database researcher might be interested in.

Another example of the way analysis of algorithms has been downgraded as an important part of computer science is a blog pointed out on Google+ by Moshe Vardi: http://feedproxy.google.com/~r/daniel-lemire/atom/~3/4bht7t0oFZc/ In this article, the author argues that one should never use a model that is not real running time on a real computer. For example, this author would not accept the O(n log n) lower bound on sorting, because it is based on counting comparisons rather than machine instructions executed. If you remember the details of sorting complexity, you know that the comparison model does not apply in some circumstances.

For example, if you are sorting n integers and they are in the range 1 to n^2, then you can sort in $O(n)$ time. Yet the sorting-by-comparisons model is still highly instructive and applies whenever you want a sorting algorithm that works on objects without some special structure like integers. For another example, I've been working a lot recently on communication complexity for MapReduce algorithms. It is generally accepted that for many problems communication cost is the bottleneck when MapReduce is used, although there are exceptions. So I think we get some instructive results and algorithm-design principles out of this analysis, even if it is not conclusive for every possible MapReduce algorithm.

It is time to recenter the pendulum. So I propose that, as reviewers of submissions to conferences or journals, we should start by asking whether the value of the proposed ideas have been analyzed for the general case. We should not accept experiments as a substitute for a more careful and general analysis, unless there really is no way to parameterize the input space suitably. And we should not accept experiments on contrived, specialized data under almost any circumstances. As authors, we should stop thinking of experiments as a substitute for analysis and deep understanding of why our algorithms work better than others that have been proposed. A little self-censorship might be a benefit to the community as well. Not every algorithm that works in some narrow window has to be published. And of course VLDB should make the question about experiments optional and include an equivalent question about whether the analysis of the algorithms is realistic.