On Experimental Algorithmics

An Interview with Catherine McGeoch and Bernard Moret

by Richard T. Snodgrass

Editor's Introduction

Computer science is often divided into two camps, systems and theory, but of course the reality is more complicated and more interesting than that. One example is the area of "experimental algorithmics," also termed "empirical algorithmics." This fascinating discipline marries algorithm analysis, which is often done with mathematical proofs, with experimentation with real programs running on real machines.

Catherine McGeoch helped initiate this approach with her 1986 Carnegie Mellon University doctoral dissertation entitled "Experimental Analysis of Algorithms." The ACM Journal of Experimental Algorithmics *was started in 1996 with Bernard Moret as founding editor; Catherine followed him in 2003. (I note in passing that this was ACM's first online-only journal. At the time, the idea of putting anything online was quite forward-looking. After all, Netscape had just been released and search engines were in their infancy.)*

My interview with these two luminaries coincided with these 25th and 15th anniversaries, examining the underpinnings of this insightful approach, examining some of its past successes, and looking towards the future.

Catherine McGeoch is the Beitzel Professor of Technology and Society at Amherst College, Massachusetts. Bernard Moret is a professor in the School of Computer and Communication Sciences at the EPFL (The Swiss Federal Institute of Technology, Lausanne, Switzerland), where he directs the Laboratory for Computational Biology and Bioinformatics.

Richard T. Snodgrass
Associate Editor

On Experimental Algorithmics

An Interview with Catherine McGeoch and Bernard Moret

by Richard T. Snodgrass

Ubiquity: The gold standard for characterizing a problem or an algorithm seems to be a statement of asymptotic complexity, obtained through a mathematical proof. So we learned several decades ago that a lower bound on sorting was between linear and quadratic (specifically, $n \log n$) and that quicksort was a way to get that. Why isn't that the end of the story? Why are we even considering experimenting with algorithms?

Bernard Moret: Three good reasons. First, on the theory side, tight bounds are often hard to obtain—until someone finds tight bounds, a thorough experimental assessment may serve well. Moreover, experimentation can help significantly with the derivation of a tight bound by hinting at the answer.

Second, tight bounds can be seriously misleading—they are derived for the general problem, yet in practice the instances to be solved tend to follow certain patterns that may make the algorithm run much faster. The classic example is the simplex algorithm for linear programming, which runs very fast in practice on just about any real-world instance, but has provably exponential worst-case behavior. In practice, moreover, even simple proportionality constants really matter; here again, there is a classic example, the suite of beautiful results in graph minors. Robertson and Seymour proved that the problem of "homeomorphic (minor) subgraph" (for given graphs G and H, can G be reduced to H through a process of vertex deletions and edge contractions?) can be solved in cubic time [1], which sounds quite attractive until you look at the proportionality constants, which include a term that is super exponential in the size of the graph H.

Third, algorithms conferences and journals offer many algorithmic improvements that knock off some slow-growing factor or trade off such a factor for a smaller one. In the development of algorithms for the classic problem of computing a minimum spanning tree in a graph, many such improvements were proposed, yet none of them produced a faster algorithm: the asymptotic running times gained relatively little and were more than offset by significantly larger overhead on graphs of reasonable sizes. In addition, the new algorithms were significantly more complex than those they claimed to improve, often relying on a succession of prior, unimplemented data structures and algorithms. Neither of these trends was of any

benefit to the practitioner. One of the goals of experimental algorithmics is to prevent such divergence.

The very tools that make theoretical analysis and tight upper bounds on running times feasible, i.e., asymptotic analysis and (a certain level of) machine-independence, also produce results that may not reflect the world of applications: constant factors are necessarily ignored, complex characteristics of real-world instances cannot be taken into account, and entrance to the world of asymptotes may lie beyond the scope of applications.

Catherine McGeoch: Bernard gives some good examples of how experimental analysis extends and complements theoretical analysis. Let me add two additional points about this relationship: First, more than just hinting at the correct theoretical bound, an experiment can directly suggest the proof strategy by making visible the underlying mechanisms that are in play. Second, in some cases the algorithm or input class is so complex (e.g., heuristics for NP-hard problems), or the platform so complicated and unpredictable (e.g., concurrent operating systems), that experimental analysis is the *only* way of determining if an algorithm works well, or even works at all.

In fact there are three main activities in algorithm research: analysis, which is about predicting performance; design, which is about finding new and better ways to solve problems; and models of computation, which is about understanding how design and analysis changes when the basic operations (defined by the platform) are modified.

In all three areas, theoretical methods necessarily make simplifying assumptions—worst-case inputs, dominant operation costs, and the RAM. Experimental methods fill in the gaps between those simplifying assumptions and real experience by incorporating interesting subclasses of inputs, constant factors and secondary costs, and more realistic machine models.

This fundamental gap between theory and practice has existed since the beginning of computing, and experiments have always been used as a bridge between the two. In recent times the gap has become an abyss, due to the increased complexity of systems and algorithms, so the bridge must be stronger and more sophisticated. Also the traffic on the bridge has become more bidirectional—we see experiments developed in the service of theory as well as of practice.

Ubiquity: What is an example where experimentation yielded insights beyond those originating from proofs of worst-case complexity?

CM: Here are three examples where experiments yielded new insights that eventually became theorems.

- Approximation algorithms for Bin Packing: Given a list of *n* weights drawn uniformly at random from the range (0,1), what is the expected asymptotic packing ratio (compared to optimal) of simple algorithms like First Fit, Best Fit, First Fit Decreasing, and Best Fit Decreasing? A series of experiments by Jon Bentley and others suggested that all four are asymptotically optimal (which contradicted previous conjectures), and also revealed patterns that inspired the arguments used in the proofs [2, 3, 4].

- Traditional worst case analysis of Dijkstra's algorithm for the shortest paths problem shows that the decrease-key operation dominates, and much design work has gone into creating data structures (such as Fibonacci heaps) that reduce the worst case bound. Experiments by Andrew Goldberg et al. suggested that, for a large category of input graphs, the decrease-key operation is rare—a property which they went on to prove [5, 6]. Those good-worst-case data structures optimized the wrong thing in many cases.

- The simple RAM model does not predict computation times on modern machines with sufficient accuracy because it does not take the memory hierarchy into account. Anthony LaMarca and Richard Ladner developed experiments to guide their design of a new two-level model of computation that captures the interactions between caches and main memory [7, 8]. They reanalyzed classic algorithms (sorting) and data structures (heaps) under the new model; their analyses are much closer to experience, and in some cases flatly contradict conventional design wisdom based on traditional analyses.

- The LaMarca and Ladner work was predated by a long and rich history of experimental and theoretical efforts—carried out by both the theory and the systems communities since around 1966—to develop two-level models of computation that describe algorithm performance in virtual memory systems. Peter Denning has a nice article that describes how theory and experiments contributed to develop our understanding of how locality of reference affects computation time [9]. A separate thread of research into cost models for I/O-bound computation has been equally fruitful.

Besides yielding new theoretical analyses and computation models, experimental algorithmics has also played a central role in algorithm and data structure design, especially in problems relating to massive data sets and memory-efficient computation.

Ubiquity: So, which is the preferred term, *experimental* algorithmics or *empirical* algorithmics? Or do these two terms mean different things?

CM: I prefer to make a distinction between "empirical," which relies primarily on observation of algorithms in realistic scenarios—analogous to a field experiment in the natural sciences—and "experimental," which emphasizes systematic manipulation of test parameters, more like a laboratory experiment. Both are valuable to the field. But I may be in the minority—most people nowadays seem to use empirical for anything involving experiments.

BM: I went back and forth on the *JEA* (*Journal of Experimental Algorithmics*) name—experimental? empirical? something else? I talked about it with many colleagues to get a sense of how it would be perceived. I got a sense that most of my colleagues in both the U.S. and Europe felt that empirical had negative connotations—what came into their minds was something like "ad hoc"—whereas experimental had neutral to positive connotations—it brought to mind something much more systematic and thorough.

This perception does not really agree with the formal definitions of the two words—Cathy has the right of it—but I felt I should go with perceptions rather than definitions and the ACM Publications Board concurred. We were taking on the rather daunting task of persuading the algorithms community that there was both value and serious critical thinking in conducting research on algorithms with actual implementation and testing; attempting in addition to that to convince them that their interpretation of the words "empirical" and "experimental" was not quite right would have made a hard task even harder.

Ubiquity: Why "Journal" rather than "Transactions," as is used by ACM much more frequently?

BM: ACM uses "Transactions" for publications focused on a particular segment of computer science, such as operating systems, databases, programming languages, etc., reserving the word "Journal" for cross-cutting publications, such as its flagship publication, the *Journal of the ACM*. *JEA* is cross-cutting in that it mixes algorithm design and analysis with software engineering, data modeling, and experimental design, and features research on the quality of algorithms applied in many different domains.

Ubiquity: How have the theory and systems communities reacted to experimental algorithmics?

CM: In the early days, with great skepticism. I remember in the late '80s chatting at a theory conference about my experimental work, and the other fellow said: "How can you call that research? Anybody can run experiments and get data—it's too easy to be called research." Twenty minutes later another colleague said: "How can you do research in that area? Nobody can learn anything useful (about asymptotes) from experiments—it's too hard." The truth is somewhere between—it is easy enough to churn out a pile of data, but not so easy to generalize the results to fundamental truths.

That response has mellowed as the field has developed, I think, but a small amount of suspicion remains that using experiments to guide analysis is "cheating"—like picking up the golf ball and walking it over to the hole, instead of whacking at it the hard way. Maybe it is a fear that analysis via computers will put theoreticians out of work.

I don't know that the systems community has expressed much reaction.

BM: Well, theoreticians just seem to prefer theorems...

SODA (the SIAM Symposium on Discrete Algorithms), the main algorithms conference, was started by David Johnson with the goal of featuring a significant amount of experimental work, work that could not get accepted at the then dominant algorithm conferences—STOC (the ACM Symposium of the Theory of Computing) and FOCS (the IEEE Symposium on the Foundations of Computer Science). Yet within a few years, SODA hardly published any application papers—it was less theoretical than STOC and FOCS only in that it focused fairly exclusively on algorithms, but it had the same orientation toward abstract algorithms, proofs of correctness, and asymptotic time bounds. In Europe, ESA (the European Symposium on Algorithms) was started as the counterpart of SODA, and again with the goal of pushing forward experimental work, but the same story played out.

Experimental work is often perceived as less profound, less intellectually challenging, and perhaps also somewhat dull because of the "bench work" involved—the kind of work one is happy to see done…by someone else! (Of course, in the natural sciences, the tables are turned—it's all about data, which gives rise to some interesting debates in bioinformatics!)

Ubiquity: Do operations research and experimental algorithmics overlap?

CM: Experimental work is an active area of operations research. For example, the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR) in summer 2010 had a special workshop on experimental methodology. The *INFORMS Journal on Computing* (formerly *ORSA Journal on Computing*) has

published experimental work on algorithms for problems of interest to the OR community since 1987. Note also that SIAM supported the ALENEX workshops from the very beginning and now publishes the ALENEX proceedings.

Most of the experimental focus in operations research (OR) has been on real-world applications of problems in combinatorial optimization, and on either heuristic search or linear programming as solution strategies. From a computer science point of view this is a fairly narrow range of applications and algorithms; the field of experimental algorithmics is much broader in scope.

BM: If you go back to experimental and empirical, note that OR, which preceded CS in this area of algorithms and applications, and which has, by and large, a much stronger commitment to applications than the computer science algorithms community, had developed guidelines for the assessment of new algorithms through computational testing, and usually referred to this type of assessment as "empirical" assessment, not "experimental" assessment. However, much of OR work is founded upon linear and integer programming and variations thereof and, while testing for these algorithmic approaches has been well developed, the computer science algorithms community works with a much broader range of problems and approaches. Thus one variant of Cathy's interpretation is that empirical does denote testing on real applications data and so is used in the more mature application areas, where approaches are more or less established; while experimental indeed denotes testing on large ranges of input parameters, often on simulated data, and thus is more common in areas where new approaches are frequently developed. The former is more characteristic of OR, the latter of computer science algorithms.

Ubiquity: What role does (or should) experimental algorithmics play in the undergraduate and graduate computer science curricula?

CM: I think general experimental methodology should play a role in computer science education at both levels, and experimental algorithmics is rich in examples and projects for students. Experiments on algorithms don't need long lead times for development, and there is much to be learned about the difficulties of gathering and interpreting performance data.

BM: There is nothing like a bit of experimentation to drive home the difference between linear and quadratic running times, to say nothing of polynomial and exponential. Students who implement and assess data structures and algorithms are immediately motivated to design better solutions. The future theoreticians in the course need no rewards beyond esthetics, but for most students motivation comes from more down-to-earth results.

Ubiquity: Looking forward, what are the major research challenges confronting experimental algorithmics? Where is this field going?

BM: A long-term failure of computer science in general and the field of algorithms in particular has been its inability to provide off-the-shelf modules with proven properties, in the style of engineering. Far too often, programmers have to code things from scratch, or to build something rather inefficient on top of existing code. The failure is not for lack of trying—the most notable example in the area of algorithms being surely LEDA, the Library of Efficient Data structures and Algorithms developed under the leadership of Kurt Mehlhorn. For some years, LEDA even became the preferred implementation medium for algorithms and data structures, yet eventually LEDA faded away.

So, one challenge I see is how to deliver modules from which programmers can build applications, such that each module has well documented (and well tested) performance. Once we can populate "shelves" with such modules, the development of algorithmic software solutions for applications should become much easier as well as much more predictable. For now, though, we are more like cathedral builders than like modern bridge or road builders: we produce one-off algorithmic solutions. Of course, much of what is needed remains to be designed by the software engineering, program correctness, and other communities within computer science.

CM: There is a good-sized algorithm engineering community, predominantly in Europe, that is working to develop systematic methodologies for bringing algorithms from pencil-and-paper abstractions to well-tuned production-quality tools. This work incorporates, besides research in algorithm efficiency, topics in interface design, specification and correctness, and software engineering. I expect that effort to continue to grow and develop.

BM: A major challenge is testing algorithms. The temptation is to test using as broad a range of instances as possible, so as to be able to report findings that apply across all or most applications. However, that comes perilously close to the asymptotic analysis conducted by a theoretician, in that it neglects the characteristics of the application data. The problem, of course, is how to model the attributes of real data. I face that issue every day in my research in computational biology. Running simulations under uniform or Gaussian distributions while assuming pairwise independence among all parameters seldom produces accurate characterizations of algorithms. (Of course, running the algorithms on a few "real" datasets does not help much either, as the "true" answers are almost always unknown in bioinformatics.) Thus I would say that modeling data so as to get at characteristics that affect the behavior of algorithms is a significant challenge for the area. We have seen such work in the

past, particularly for sorting lists of numbers or computing simple properties of graphs; recent work on phase transitions in the space of instances also points in that direction.

CM: Another looming question is how to cope with new design and analysis problems relating to low-level parallelism. Every desktop and laptop nowadays contains two to 16 cores, and the algorithms community does not have a good handle on how to design and analyze algorithms for these new platforms. I think that area will, of necessity, have to grow very soon. The experimental algorithmics community is well positioned to lead the way.

Problems in big data will continue to get bigger and there is still a lot of research work to be done on heuristics and approximation algorithms for NP-hard problems.

Finally there is a shortage of statistical and data analysis techniques for answering the types of questions that algorithm experimenters like to ask. For example: there is no standard data analysis technique for analyzing a data set to find—or bound—the leading term in the (unknown) function that produced the data. It is not even known how to test whether a data sample supports a conjecture of polynomial versus exponential growth. Little is known about how to design experiments to analyze functional growth in data. Nobody knows if different statistical analyses should be used on trace data versus independent samples.

Ubiquity: What are the implications for the future of computing?

CM: I hope that continued progress in experimental algorithmics will enable the theory community to build much stronger connections to other areas of computing research.

In my ideal future we will not see a dichotomy—theory versus practice—but rather a continuum of algorithm research along a scale between abstract and concrete. For example, a single algorithm may be analyzed in terms of an asymptotic bound on the dominant cost; or exact counts of dominant and secondary operations; or instruction counts on generic platforms; or clock ticks in a specific system. In the past nearly all algorithm research was carried out in either the first or last of these scenarios. Experimental algorithmics can make important contributions to research in the middle areas as well as at the ends.

BM: In the pioneering era of computing (the 1960s and early 1970s), theoretical and experimental algorithmics were much the same, and thorough experimental assessments were common across all of computer science: systems researchers compared competing algorithms for job scheduling by running them in real systems, data structure designers tailored their structures to the architecture of the machine, and there was considerable peer pressure to implement and test any new designs. Along the way, the theory and experiment became

decoupled, perhaps because of enormous complexities in setting up meaningful experiments. We are moving now toward reintegration of these two aspects of our field. I doubt that we can return to a full integration unless some new advances bring about a huge simplification of the field, but full integration is not required as long as information and motivation keeps flowing between the diverse communities. Experimental algorithmics is one of the processes at work to keep such communication channels open.

References

1. Neil Robertson and Paul Seymour. Graph Minors XIII: The disjoint paths problem. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 65–110.

2. J.L. Bentley, D.S. Johnson, F.T. Leighton, and C.C. McGeoch. An experimental study of bin packing. In *Proceedings of the 21st Allerton Conference on Computing, Control, and Communication*. (1983).

3. J.L. Bentley, D.S. Johnson, F.T. Leighton, C.C. McGeoch, and L.A. McGeoch. Some unexpected expected behavior results for bin packing. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*. (1984).

4. Peter W. Shor. The average-case analysis of some on-line algorithms for bin packing. *Combinatorica* 6,2 (1986), 179-200.

5. Boris Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming* 73 (1993), 129–174.

6. Andrew V. Goldberg and Robert E. Tarjan. Expected performance of Dijkstra's shortest path algorithm. *NEC Research Institute Report* (1996).

7. Anthony LaMarca and Richard E. Ladner. The influence of caches on the performance of sorting. SODA (1997).

8. Anthony LaMarca and Richard E. Ladner. The influence of caches on the performance of heaps. *ACM Journal of Experimental Algorithmics* 1 (1996).

9. Peter J. Denning. The locality principle. *Commun. ACM* 48,7 (2005), 19–24.

About the Author

Richard Snodgrass is a Professor of Computer Science at the University of Arizona, working in the areas of ergalics and temporal databases.

DOI: 10.1145/2555235.2555238

Appendix

"Resources for Experimental Algorithmics"
By Catherine McGeoch and Bernard Moret

The discipline of experimental algorithmics has been an active one since about 1990. There were pockets of active research prior to that, such as the works of Jon Bentley and David Johnson at Bell Labs, Bernard Moret and Henry Shapiro at the University of New Mexico, and Catherine McGeoch at Carnegie Mellon University.

JOURNALS

The ACM *Journal of Experimental Algorithmics* was started in 1995. Bernard Moret was the founding Editor-in-Chief, Catherine McGeoch followed him in 2003, and Guiseppe Italiano has been EIC since 2008.

WORKSHOPS

In 1990 the first ACM-SIAM Symposium on Data Structures and Algorithms (SODA) was organized by David Johnson. The call for papers explicitly invited "analytical or experimental" analyses, which may be "theoretical or based on real datasets." Also in 1990, the first DIMACS Implementation Challenge was co-organized by David Johnson and Catherine McGeoch. DIMACS Challenges are year-long cooperative research efforts in experimental algorithmics.

The Workshop on Algorithm Engineering (WAE) was founded in 1997 in Venice, Italy by Giuseppe Italiano and ran for five years (Saarbrucken, Germany, 1998, London, England, 1999, Saarbrucken, Germany, 2000, and Aarhus, Denmark, 2001) before being reabsorbed by the annual European Symposium on Algorithms (ESA), as the "Engineering and Applications" track. ESA set up a separate track for WAE, with its own program committee separate from that of the (larger) theoretical track. The original goal was to have the conference site alternate between Europe and the U.S., but ultimately WAE was very much a European effort; Moret was on its steering committee as the U.S. representative. The 1999-2001 proceedings were published by Springer in its *Lecture Notes in Computer Science* (numbers 1668, 1982, and 2141).

The Workshop on Algorithms and Experiments (ALEX) was organized by Roberto Battiti in 1998, also in Italy. This inspired the Workshop on Algorithm Engineering and Experiments (ALENEX), which began in 1999 in the U.S., with Catherine McGeoch and Mike Goodrich as its co-founders. It has been held in Baltimore, MD in 1999, San Francisco, CA in 2000, Washington, DC in 2001, San Francisco in 2002, Baltimore, MD in 2003, New Orleans, LA in 2004, Vancouver, BC, Canada in 2005, Miami, FL in 2006, New Orleans, LA, 2007, San Francisco, CA, 2008, New York City, NY, 2009, and Austin, TX, 2010. Starting in 2003 the proceedings have been published by SIAM; the first three were Springer LNCS (numbers 1619, 2153, and 2409). This conference continues to this day, in a format with two PC co-chairs—one North American and one European. It has been steered by a very large steering committee (all past PC chairs), but really by a few people who put in the work to identify new PC chairs. It runs as a one-day meeting before the SODA meeting. In the last few years, it has become an official SIAM meeting—a necessary step, as it was a real burden running a small workshop without any backing organization. So ALENEX is very much a North American meeting, the counterpart to its European sibling WAE/ESA Applied Track, just like SODA and ESA are the respective main algorithm meetings on each continent.

The International Workshop on Efficient Algorithms (WEA) was organized by Klaus Jansen and Evipidis Bampis in 2001. In 2003 this became the International Workshop on Experimental and Efficient Algorithms (WEA) coordinated by José Rolim. It was held in Ascona, Switzerland in 2003, Angra dos Reis, Brazil in 2004, Santorini Island, Greece in 2005, Cala Galdana, Menorca Island, Spain in 2006, Rome, Italy in 2007 and Provincetown, MA in 2008. In 2009 it became the Symposium on Experimental Algorithms (SEA), held in Dortmund, Germany in 2009, in Ischia Island, Naples, Italy in 2010, and in Kolimpari, Chania, Crete, Greece, in 2011. The proceedings are all published by Springer LNCS, numbers 2647, 3059, 3503, 4007, 4525, 5038, 5526, 6049, and 6630.

Bernard Moret started WABI (Workshop on Algorithms in Bioinformatics) in 2001 in Denmark, in the spirit of the WAE/ALENEX meetings. It has been held in Aarhus, Denmark in 2001, Rome, Italy in 2002, Budapest, Hungary in 2003, Bergen, Norway in 2004, Mallorca, Span in 2005, Zurich, Switzerland in 2005, Philadelphia, PA in 2007, Karlsruhe, Germany in 2008, Philadelphia, PA in 2009, and Liverpool, England in 2010. From the beginning, the proceedings have been published by Springer LNCS (numbers 2149, 2452, 2812, 3240, 3692, 4175, 4645, 5251, 5724, and 6293). WABI remains a mix of pure algorithms and algorithm engineering within the area of computational biology—an area in which most conferences initially oriented toward algorithms (RECOMB, APBC, ISBRA) have drifted more and more toward biological applications. WABI remains unaffiliated so far, but usually runs with ESA (it has run twice in the U.S. so far, with a somewhat larger audience); it is larger than either ESA Applied Track or ALENEX, with about two to three times the number of papers—typically, about 100 submissions and 30-35

acceptances. Like ALENEX, it has two PC co-chairs every year—in this case one from North America and one from Europe, one being more biology-oriented and one more CS-oriented.

DIMACS IMPLEMENTATION CHALLENGES

The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) has sponsored Implementation Challenges since 1990. Each Challenge is a coordinated year-long research effort to develop efficient algorithms for a small number of problem areas; at the end of each Challenge, a workshop is held at the DIMACS center on the campus of Rutgers University, and proceedings are published by AMS as part of the DIMACS Series in Discrete Mathematics and Theoretical Computer Science. A software repository at the DIMACS website contains instance generators and solvers for public use. The first Challenge 1990-1991 was co-organized by David Johnson and Catherine McGeoch, and covered algorithms for Network Flows and Matching. The second Challenge 1992-1993, organized by Michael Trick, considered Maximum Cliques, Graph Coloring, and Satisfiability. The third Challenge 1993-1994, organized by Sandeep Bhatt, looked at Parallel Algorithms for Combinatorial Problems. The fourth Challenge 1994-1995, organized by Martin Vingron, considered Fragment Assembly and Genome Rearrangements. The fifth Challenge in 1995-1996, organized by Catherine McGeoch, covered Priority Queues, Dictionaries, and Multi-Dimensional Point Sets. The topic of the sixth Challenge in 1998 was Near Neighbor Searching; it was organized by Michael Goldwasser. The seventh Challenge in 2000, which was organized by David Johnson, Gabor Pataki, and Farid Alizadeh, considered Semidefinite and related Optimization Problems. In 2001, the eighth Challenge, organized by David Johnson, Lyle McGeoch, Fred Glover, and Cesar Rego, looked at algorithms for the Traveling Salesman Problem. The ninth Challenge, in 2006, was organized by Camil Demetrescu, Andrew Goldberg, and David Johnson, and considered the Shortest Path Problem. Most recently, the tenth Challenge in 2011 was organized by David Bader, Peter Sanders, and Dorothea Wagner, and considered algorithms for Graph Partitioning and Graph Clustering. Contact David Johnson at AT&T Labs-Research if you are interested in organizing an Implementation Challenge.

DAGSTUHLs

There have been four Dagstuhl workshops on Algorithm Engineering. The first in September 2000 was entitled “Experimental Algorithmics” and was organized by Rudolf Fleischer, Bernard Moret, and Erik Schmidt. The second one was in September 2002, with the same crew of organizers, joined by Jon Bentley. The third in September 2006 was entitled “Algorithm Engineering” and was organized by Matthias Müller-Hanneman and Stefan Schirra. The fourth in June 2010 was also on “Algorithm Engineering” and was organized by Guiseppe F. Italiano, David S. Johnson, Petra Mutzel, and Peter Sanders.

BOOKS

Paul Cohen, *Empirical Methods for Artificial Intelligence*, MIT Press 1995. This is a college-level introductory textbook on statistics and data analysis, with numerous examples drawn from experiments on algorithms.

L. Rapanotti, *Algorithm Engineering for Integral and Dynamic Problems,* Amsterdam: Gordon & Breach, Abingdon: Marston, 2001.

Rudolf Fleischer, Bernard M. E. Moret, and Erik M Schmidt, Experimental Algorithmics: From algorithm design to robust and efficient software, *Lecture Notes in Computer Science, 2547*, Springer, 2002. (From the 2000 Dagstuhl)

Matthias Müller-Hanneman and Stefan Schirra, eds. *Algorithm Engineering—Bridging the Gap Between Algorithm Theory and Practice*, *Lecture Notes in Computer Science 5971*, Springer, 2010. (This book, from the 2006 Dagstuhl seminar, does not read like a proceedings, but rather like a tutorial. The book editors assigned groups of graduate students to research each topic, give a presentation at the seminar, and write a survey article that became a chapter in the book. Some chapters and bridge material were written by the editors.)

Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, eds., *Experimental Methods for the Analysis of Optimization Algorithms*, Springer 2010.

Catherine McGeoch, *A Guide to Experimental Algorithmics*, Cambridge University Press, to appear 2011.

WEBSITE

Bibliography on Experimental Methods for the Analysis of Search and Optimization Algorithms; Marco Chiarandini, University of Southern Denmark, Accessed August 2011.

Reprinted with Permission from* Ubiquity *August 2011
DOI: 10.1145/2015996.2015997

