Engineering and Visualizing Algorithms*

Camil Demetrescu[1], Irene Finocchi[2], and Giuseppe F. Italiano[2]

[1] Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Salaria 113, 00198 Roma, Italy. demetres@dis.uniroma1.it,
http://www.dis.uniroma1.it/~demetres/

[2] Dipartimento di Informatica, Sistemi e Produzione,
Università di Roma "Tor Vergata",
Via del Politecnico 1, 00133 Roma, Italy.
{finocchi,italiano}@disp.uniroma2.it,
http://www.info.uniroma2.it/~{finocchi,italiano}/

Abstract. We discuss some relevant issues in Algorithm Engineering, focussing on the interplay between theory and practice, and showing how it can integrate and reinforce the traditional theoretical approaches to the design and analysis of algorithms and data structures, while devising methodologies and tools for developing and engineering efficient algorithmic codes.

1 Introduction

The whole process of designing, analyzing, implementing, tuning, debugging and experimentally evaluating algorithms can be referred to as *Algorithm Engineering*. Algorithm Engineering views algorithmics also as an engineering discipline rather than a purely mathematical discipline. Implementing algorithms and engineering algorithmic codes is a key step for the transfer of algorithmic technology, which often requires a high-level of expertise, to different and broader communities, and for its effective deployment in industry and real applications. Moreover, experiments often raise new conjectures and theoretical questions, opening unexplored research directions that may lead to further theoretical improvements and eventually to more practical algorithms. Theoretical breakthroughs that answer fundamental algorithmic questions are a crucial step towards the solution of a problem. However, they often lead to algorithms that are far from efficient implementations, raising the question of whether more practical solutions exist. We believe that Algorithm Engineering should take into account the *whole* process, from the early design stage to the realization of efficient implementations.

* Work partially supported by the IST Programmes of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-2001-33555 (COSIN), and by the Italian Ministry of University and Scientific Research (Project "ALINWEB: Algorithmics for Internet and the Web").

An important aspect of algorithm engineering, usually referred to as *Experimental Algorithmics*, is related to performing empirical studies for comparing actual relative performance of algorithms so as to study their amenability for use in specific applications. This may lead to the discovery of algorithm separators, i.e., families of problem instances for which the performances of solving algorithms are clearly different, and to identifying and collecting problem instances from the real world. Other important results of empirical investigations include assessing heuristics for hard problems, characterizing the asymptotic behavior of complex algorithms, discovering the speed-up achieved by parallel algorithms and studying the effects of the memory hierarchy and of communication on real machines, thus helping in predicting performance and finding bottlenecks in real systems. Another goal of algorithm engineering is to define standard methodologies and realistic computational models. For instance, there is more and more interest in defining models for the memory hierarchy and for Web algorithmics. Issues related to Web algorithmics are received a lot of attention, as the Internet is nowadays a primary motivation for several problems: security infrastructure, Web caching, Internet searching and information retrieval are just a few of the hot topics. Devising realistic models for the Internet and Web graphs is thus essential for testing the algorithmic solutions proposed in this settings.

2 Algorithm Engineering and Visualization

Algorithms must be implemented and tested in order to have a practical impact. Experiments can help measure practical indicators, such as implementation constant factors, real-life bottlenecks, locality of references, cache effects and communication complexity, that may be extremely difficult to predict theoretically. We remark here that locality and cache effects are particularly important for graph algorithms: graphs are typically de-localized and pointer-based data structures. Many clear examples of the value of the experimentation for graph algorithms are addressed in the literature: among them, the implementation issues of the push-relabel algorithm for the maximum flow problem by Goldberg and Tarjan [9] stand out.

Unfortunately, as in any empirical science, it may be sometimes difficult to draw general conclusions about algorithms from experiments. Towards this aim, some researchers have proposed accurate and comprehensive guidelines on different aspects of the empirical evaluation of algorithms maturated from their own experience in the field (see, for example [1, 14, 15, 18]). The interested reader may find in [17] an annotated bibliography of experimental algorithmics sources addressing methodology, tools and techniques.

Among the tools useful in algorithm engineering, we cite visualization systems, which exploit interactive graphics to enhance the development, presentation, and understanding of computer programs [21]. Thanks to the capability of conveying a large amount of information in a compact form that is easily perceivable by a human observer, visualization systems can help developers gain insight about algorithms, test implementation weaknesses, and tune suitable heuristics for improving the practical performances of algorithmic codes. Some examples of this kind of usage are described in [11].

Systems for algorithm animation have matured significantly since the rise of modern computer graphic interfaces and dozens of algorithm animation systems have been developed in the last two decades [2–8, 10, 12, 16, 19, 20, 22]. For a comprehensive survey we refer the interested reader to [13, 21] and to the references therein. In the following we limit to discuss the features of algorithm visualization systems that appear to be most appealing for their deployment in engineering algorithms.

From the viewpoint of the algorithm developer, it is desirable to rely on systems that offer visualizations at a *high level of abstraction*. Namely, one would be more interested in visualizing the behavior of a complex data structure, such as a graph, than in obtaining a particular value of a given pointer.

Fast prototyping of visualizations is another fundamental issue: algorithm designers should be allowed to create visualization from the source code at hand with little effort and without heavy modifications. At this aim, *reusability* of visualization code could be of substantial help in speeding up the time required to produce a running animation.

One of the most important aspects of algorithm engineering is the development of *libraries*. It is thus quite natural to try to interface visualization tools to algorithmic software libraries: libraries should offer default visualizations of algorithms and data structures that can be refined and customized by developers for specific purposes.

Software visualization tools should be able to animate *not just "toy programs"*, but significantly complex algorithmic codes, and to test their behavior on large data sets. Unfortunately, even those systems well suited for large information spaces often lack advanced navigation techniques and methods to alleviate the screen bottleneck. Finding a solution to this kind of limitations is nowadays a challenge.

Advanced debuggers take little advantage of sophisticated graphical displays, even in commercial software development environments. Nevertheless, software visualization tools may be very beneficial in addressing problems such as finding memory leaks, understanding anomalous program behavior, and studying performance. In particular, environments that provide interpreted execution may more easily integrate advanced facilities in support to *debugging and performance monitoring*, and many recent systems attempt at exploring this research direction.

There is a general consensus that algorithm visualization systems can strongly benefit from the potentialities offered by the *World Wide Web*. Indeed, the use of the Web for easy communication, education, and distance learning can be naturally considered a valid support for improving the cooperation between students and instructors, and between algorithm engineers.

3 Concluding Remarks

Algorithm engineering refers to the process of designing, analyzing, implementing, tuning, debugging, and experimentally evaluating algorithms. It refines and reinforces the traditional theoretical approach with experimental studies, fitting the general models and techniques used by theoreticians to actual existing machines and leading to robust and efficient implementations of algorithms and data structures that can be used by non-experts. Trends in engineering algorithms include the following. First of all, programs should be considered as first class citizens: going from efficient algorithms to programs is not a trivial task and the development of algorithmic libraries may be helpful for this. Furthermore, in may applications seconds matter, and thus experimentation can add new insights to theoretical analyses. Moreover, current computing machines are far away from RAMs: if algorithms are to have a practical utility, issues such as effects of the memory hierarchy (cache, external memory), implications of communication complexity, numerical precision must be considered. Machine architecture, compiler optimization, operating system, programming language are just a few of the technical issues that may substantially affect the execution performance. Algorithmic software libraries, tools for algorithm visualization, program checkers, generators of test sets for experimenting with algorithms may be of great help throughout this process.

References

1. R. Anderson. The role of experiment in the theory of algorithms. In *Proceedings of the 5th DIMACS Challenge Workshop*, 1996. Available over the Internet at the URL: `http://www.cs.amherst.edu/~dsj/methday.html`.
2. J.E. Baker, I. Cruz, G. Liotta, and R. Tamassia. A New Model for Algorithm Animation over the WWW. ACM Computing Surveys, 27(4):568–572, 1996.
3. J.E. Baker, I. Cruz, G. Liotta, and R. Tamassia. Animating Geometric Algorithms over the Web. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages C3–C4, 1996.
4. J.E. Baker, I. Cruz, G. Liotta, and R. Tamassia. The Mocha Algorithm Animation System. In *Proceedings of the 1996 ACM Workshop on Advanced Visual Interfaces*, pages 248–250, 1996.
5. R.S. Baker, M. Boilen, M.T. Goodrich, R. Tamassia, and B. Stibel. Testers and visualizers for teaching data structures. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, 31, 1999.
6. M.H. Brown. *Algorithm Animation.* MIT Press, Cambridge, MA, 1988.
7. M.H. Brown. Zeus: a System for Algorithm Animation and Multi-View Editing. In *Proceedings of the 7-th IEEE Workshop on Visual Languages*, pages 4–9, 1991.
8. G. Cattaneo, G.F. Italiano, and U. Ferraro-Petrillo. CATAI: Concurrent Algorithms and Data Types Animation over the Internet. *Journal of Visual Languages and Computing*, 13(4):391–419, 2002. System Home Page: `http://isis.dia.unisa.it/catai/`.
9. B.V. Cherkassky and A.V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

10. P. Crescenzi, C. Demetrescu, I. Finocchi, and R. Petreschi. Reversible Execution and Visualization of Programs with Leonardo. *Journal of Visual Languages and Computing*, 11(2), 2000. System home page: `http://www.dis.uniroma1.it/~demetres/Leonardo/`.
11. C. Demetrescu, I. Finocchi, G.F. Italiano, and S. Naeher. Visualization in algorithm engineering: Tools and techniques. In *Dagstuhl Seminar on Experimental Algorithmics 00371*. Springer Verlag, 2001.
12. C. Demetrescu, I. Finocchi, and G. Liotta. Visualizing Algorithms over the Web with the Publication-driven Approach. In *Proc. of the 4-th Workshop on Algorithm Engineering (WAE'00)*, LNCS 1982, pages 147–158, 2000.
13. S. Diehl. *Software Visualization*. LNCS 2269. Springer Verlag, 2001.
14. A.V. Goldberg. Selecting problems for algorithm evaluation. In *Proc. 3-rd Workshop on Algorithm Engineering (WAE 99), LNCS 1668*, pages 1–11, 1999.
15. D. Johnson. A theoretician's guide to the experimental analysis of algorithms. In *Proceedings of the 5th DIMACS Challenge Workshop*, 1996. Available over the Internet at the URL: `http://www.cs.amherst.edu/~dsj/methday.html`.
16. A. Malony and D. Reed. Visualizing Parallel Computer System Performance. In M. Simmons, R. Koskela, and I. Bucher, editors, *Instrumentation for Future Parallel Computing Systems*, pages 59–90. ACM Press, 1999.
17. C. McGeoch. A bibliography of algorithm experimentation. In *Proceedings of the 5th DIMACS Challenge Workshop*, 1996. Available over the Internet at the URL: `http://www.cs.amherst.edu/~dsj/methday.html`.
18. B.M.E. Moret. Towards a discipline of experimental algorithmics. In *Proceedings of the 5th DIMACS Challenge Workshop*, 1996. Available over the Internet at the URL: `http://www.cs.amherst.edu/~dsj/methday.html`.
19. G.C. Roman, K.C. Cox, C.D. Wilcox, and J.Y Plun. PAVANE: a System for Declarative Visualization of Concurrent Computations. *Journal of Visual Languages and Computing*, 3:161–193, 1992.
20. J.T. Stasko. Animating Algorithms with X-TANGO. *SIGACT News*, 23(2):67–71, 1992.
21. J.T. Stasko, J. Domingue, M.H. Brown, and B.A. Price. *Software Visualization: Programming as a Multimedia Experience*. MIT Press, Cambridge, MA, 1997.
22. A. Tal and D. Dobkin. Visualization of Geometric Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):194–204, 1995.