Chapter 10. Challenges in Algorithm Engineering

Matthias Müller-Hannemann and Stefan Schirra

This final book chapter is meant as a brief reflection on the current status of Algorithm Engineering and its future development. It is devoted to the many challenges this discipline has to face. By challenges we mean things that are worthy to invest a significant research effort, and working on these problems promises a high potential impact.

In early 2007, the authors made a poll among colleagues questioning about the most important challenges for Algorithm Engineering. We also asked them about future trends and developments for the discipline they envision. The answers we obtained covered a broad range of issues which we tried to integrate into the following discussion and overview. We acknowledge thankfully contributions by David Bader, Ulrik Brandes, Hervé Brönniman, Dan Halperin, Riko Jacob, Michael Jünger, Ernst Mayr, Cathy McGeoch, Kurt Mehlhorn, Petra Mutzel, Stefan Näher, Hartmut Noltemeier, Knut Reinert, Peter Sanders, Anita Schöbel, Steve Skiena, Anand Srivastav, Raimund Seidel, Berthold Vöcking, and Ingo Wegener. Another valuable source for challenges is the grant proposal by Mehlhorn et al. for the DFG funding initiative on Algorithm Engineering (www.algorithm-engineering.de). Nevertheless, the following compilation describes the personal view of the authors in first place.

We start our discussion with general remarks on the discipline, and then list challenges along the different phases of the Algorithm Engineering cycle. Thus, the order of topics should not be interpreted as a ranking with respect to importance.

10.1 Challenges for the Algorithm Engineering Discipline

According to the French philosopher Bruno Latour, much science-in-the-making appears as art until it becomes settled science. Algorithm Engineering is a quite young and evolving new discipline. Therefore, it is quite natural that it has to face a number of challenges with respect to its own development. The main challenge is probably to further establish Algorithm Engineering as a scientific discipline in algorithmics and more generally in computer science.

In algorithmics, theoretical and experimental research have been to a large extent separated since the 1970s and 1980s, and experimental work in algorithmics often does not yet get the credit it deserves, although experimentation is an integral part of the process of taking theoretical results to practice. Experimental algorithmics is still often only considered as a substitute where theoretical analysis fails, not as a complementary method to better study what is best suited to solve real-world problems at hand in practice. Still, driven by the need for practical solutions Algorithm Engineering has entered algorithmics over the past decade. Besides classical asymptotic algorithm analysis, there are now more and
more field experiments that measure runtimes on real-world problem instances in order to get more precise information about program performance in practice. But Algorithm Engineering is more than that, e.g. it can also lead to new algorithmic insights.

The most general and ambitious goal of Algorithm Engineering is to close the gap between theory and practice in algorithmics. This by itself is obviously a major challenge as Algorithm Engineering has to make use of the algorithmic knowledge developed in theory and thus has to turn sophisticated methods into (re)usable software. Regarding runtime prediction the gap between asymptotic unit cost analysis and performance on real computers is ever increasing with modern system architectures that have multicore processors and multilevel caches. Engineering practical efficient algorithms for multicore, parallel, and distributed systems is another major general challenge in Algorithm Engineering these days.

10.1.1 Realistic Hardware Models

The demand for experimentation in Algorithm Engineering exists because asymptotic analysis and the models of computation used in theory do not allow for predicting the actual performance in practice accurately. The gap between theory and practice in algorithmics is partially due to a gap between models of computation, like a random access machine with uniform cost measures, and existent computers today. Thus we need more useful models. We need models that are closer to the actual hardware and allow for more accurate performance prediction. Ideally, such better models are still simple enough for design and analysis, but let the algorithm designers tune their algorithms prior to the experimentation phase or might even render the experimentation phase obsolete.

Modern architectures are particularly challenging. With respect to memory hierarchies, the cache-aware and cache-oblivious models are important steps forwards, but they have their limitations as well and still must prove impact on practice.

Use of flash memory is one of the latest developments. Flash memory will either completely replace magnetic hard disks or become at least an additional secondary storage in the near future. Flash memory is similar to RAM with respect to its ability of fast random reads, but also similar to hard disks as a block based device with slow write operations. Since write operations require the erasure of a whole block, and erased blocks wear out, erasures should be spread out almost evenly over the blocks to extend the life time of the chip. This is called wear leveling and requires new algorithmic designs, something like “flash-aware” or “flash-oblivious” algorithms. The latter is particularly desired since flash technology still changes very rapidly. Algorithmic research in this area has just started.

The key challenge today is the design and analysis of efficient, high-performance algorithms for multicore and parallel processors. The industry has adopted multicore as a mechanism to continue to leverage Moore’s Law, and parallel algorithm design now moves from a special niche to the mainstream. Yet the masses of