Preface

Algorithm Engineering is a methodology for algorithmic research that combines theory with implementation and experimentation in order to obtain better algorithms with high practical impact. Traditionally, the study of algorithms was dominated by mathematical (worst-case) analysis. In Algorithm Engineering, algorithms are also *implemented* and experiments conducted in a *systematic way*, sometimes resembling the experimentation processes known from fields such as biology, chemistry, or physics. This helps in counteracting an otherwise growing gap between theory and practice. The possible benefits are manifold:

- We can *identify practically good algorithms*, even where theoretical analysis is lacking or does not make an accurate enough statement. This can lead to successful algorithms being recognized and put into practice even when they look mathematically inaccessible. It can also lead to theoretically appealing algorithms being recognized as less useful in practice. For example, suppose that we have one algorithm with a quadratic worst-case running time and another with a cubic bound, both proven by theory. Which of the two is faster in practice? This cannot be answered by merely looking at the degrees of the polynomials.
- Experimentation can be supportive of design and analysis. It can help to identify bottlenecks that could be eliminated by a re-design. It can help to build and to test falsifiable hypotheses, e.g., whether a particular algorithm after a modification that makes it faster will still deliver the same solution quality. This creates a *feedback loop* where each particular sub-methodology benefits from the others.
- Observations of good performance in practice of an algorithm for which we only know unsatisfactory theoretical guarantees have sparked new types of theoretical algorithm analyses. In particular, *smoothed analysis* instead of the traditional worst-case analysis has been shown to be successful. In smoothed analysis, the input is first chosen as in worst-case analysis (i.e., we consider all possible inputs) but then small random perturbations are performed. We then conduct an average-case analysis using the resulting distribution on the set of instances.
- The requirement to implement and to experimentally evaluate algorithms motivates algorithm designers to think about practicability and to consider *more realistic computational models*. New possibilities and their restrictions given by modern hardware are moved into focus, such as memory hierarchies and parallelism.
- Testing algorithms on real-world input stimulates the *exchange and cooperation* between the algorithmics community and other fields where algorithms are used as tools.

This volume reviews important algorithmic developments and results that were made possible or were supported by Algorithm Engineering. Chapters are ordered alphabetically by first author. This work emerged from the Priority Programme "Algorithm Engineering" (1307) funded by the German Research Foundation (DFG),

which started in 2007 and lasted six years. In total, 28 projects received funding through this program. In addition there were six associated projects. We gratefully acknowledge this support.

Each submission for this volume was peer-reviewed. We sincerely thank the authors and the reviewers for their work, diligence, and cooperation.

In total, we have 12 chapters, including extensive surveys and case studies:

Chapter 1 A simple but powerful stochastic local search algorithm for the SAT problem is presented and analyzed. Experiments are used for tuning and for comparison with other algorithms. It is concluded that when flipping a variable, it is more important to pay attention to the number of newly unsatisfied clauses than to the number of newly satisfied ones.

Chapter 2 is a survey on practical algorithms for routing in transportation networks, including road networks, schedule-based public transport networks, as well as multimodal scenarios. Experiments show that it is possible to find good journeys within milliseconds in large-scale networks. Several of the described approaches have been included in mainstream production systems.

Chapter 3 surveys different ways to theoretically analyze the k-means clustering algorithm. Several of the theoretical activities, e.g., smoothed analysis, were motivated by observations in experiments.

Chapter 4 surveys practical algorithms for balanced graph partitioning. A large variety of different approaches are presented, and implementation aspects and benchmarking are discussed.

Chapter 5 In randomized and derandomized rounding, for many applications it is required that the solution satisfies certain constraints with probability one. In order to obtain such solutions, there exist two very different algorithmic approaches, which, however, have very similar theoretical properties. This chapter surveys theoretical foundations, experimental studies, and applications for those two original approaches and new ones derived from them.

Chapter 6 is a review of external-memory search for state space problems, giving detailed descriptions of algorithms and data structures, complemented by concrete examples. Implementation on a GPU is discussed and speedups are substantiated by experiment.

Chapter 7 presents a framework for the development and evaluation of real-time rendering algorithms. A central concept is a meta rendering algorithm that automatically selects an algorithm for the visualization of highly complex scenes.

Chapter 8 applies the Algorithm Engineering cycle of design, analysis, implementation, and experimentation to robust optimization. In such problems, the exact data is not known but bounded by a set of possible realizations. The importance of considering real-world applications is demonstrated.

Chapter 9 gives a survey on concepts and algorithms for finding clusters in networks that change over time. Data sets for experimentation, comprised of real-world and synthetic data, are thoroughly discussed.

Chapter 10 Many industrial production planning problems have both sequencing and allocation aspects. This chapter describes and experimentally evaluates a framework based on genetic algorithms that can smoothly integrate both aspects.

Chapter 11 A streaming algorithm for the bipartite matching problem is gradually improved using experimentation. This finally results in a version that inherits all the good theoretical properties of the original version while being much faster in practice.

Chapter 12 is a survey and a unified, extensive experimental comparison of algorithms for the art gallery problem – a classic and important problem from computational geometry. Moreover, a new and superior implementation is presented, which combines the best techniques identified in previous approaches.

February 2016

Lasse Kliemann
Peter Sanders

http://www.springer.com/978-3-319-49486-9

Algorithm Engineering
Selected Results and Surveys
Kliemann, L.; Sanders, P. (Eds.)
2016, X, 419 p. 68 illus., Softcover
ISBN: 978-3-319-49486-9