Computation and Complexity

Gerhard J. Woeginger(✉)

Department of Computer Science, RWTH Aachen, 52074 Aachen, Germany
woeginger@algo.rwth-aachen.de

Abstract. Algorithmics, computation, optimization, complexity, combinatorics and knowledge representation are closely related sub-areas of Theoretical Computer Science. The following summary presents short descriptions of the twelve chapters in this topical part.

This first part of the book covers a wide range of topics that may roughly be summarized under the words *"computation"* and *"complexity"*. The choice of topics reflects the massive developments in computer science over recent decades. Many research areas that used to be outside of traditional computer science have been conquered and attacked with computer science tools. For example, classical Euclidean geometry has led to the area *computational geometry*, classical graph theory has led to *algorithmic graph theory*, biology gave us the area of *computational biology*, from the social sciences we got *computational social choice*, and economics delivered the areas of *algorithmic game theory* and *computational economics*. Computer science has always been very successful in modelling communication systems, for example *wireless networks*. One of the biggest challenges in neuroscience consists in understanding how the human brain works and how the *brain performs computations*. The internet (as a gigantic decentralized computing system) has led to the areas *data mining* and *knowledge harvesting*.

The first part of the book deals with some of these challenges and trends, and the following twelve chapters analyze some aspects of these developments. We present short descriptions of these chapters.

The chapter **"Some Estimated Likelihoods for Computational Complexity"** by Williams [16] addresses some of the most fundamental open problems in computational complexity theory. Of course, we would like to know the answer to the P versus NP question (Is $\mathsf{P} = \mathsf{NP}$?). A slightly easier problem asks whether $\mathsf{P} = \mathsf{PSPACE}$ (this statement should at least be easier to disprove than $\mathsf{P} = \mathsf{NP}$, as NP$\subseteq$PSPACE holds). Other central open questions in complexity theory concern the so-called Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [9], the Strong Exponential Time Hypothesis (SETH), and the Nondeterministic Strong Exponential Time Hypothesis (NSETH), which all form strengthenings of the statement $\mathsf{P} \neq \mathsf{NP}$. In his chapter, Ryan Williams states concrete probabilities with which he believes that the various open problems have positive answers: $\mathsf{P} \neq \mathsf{NP}$ with probability 80%, ETH should hold with probability 70%, SETH with probability 25%, and so on. The body of the chapter deals with the reasons why and how Ryan arrived at these probabilities, with technical possibilities and impossibilities, and many other things.

B. Steffen and G. Woeginger (Eds.): Computing and Software Science, LNCS 10000, pp. 3–8, 2019.
https://doi.org/10.1007/978-3-319-91908-9_1

The chapter **"Computing in Combinatorial Optimization"** by Cook [5] summarizes some of the history of algorithms for combinatorial optimization problems. One of the most prominent problems in this area is the Travelling Salesman Problem (TSP): Given a finite number of points with pairwise distances, find a shortest path connecting all points. William Cook traces the problem back to the year 1930, when Karl Menger described the problem in a colloquium held in Vienna; Menger used the German word "Botenproblem" ("postman problem") for the TSP, as the problem is faced in practice by every postman. Starting from the TSP, the chapter walks through a variety of other fundamental problems and algorithms. The reader learns many surprising facts on the history of the assignment problem, the Hungarian algorithm, dynamic programming, linear programming, cutting planes, matchings, the simplex method, CPLEX, and the reader meets Kurt Gödel, Julia Robinson, Richard Karp, George Dantzig, Michel Balinski, Richard Bellman, Jack Edmonds, Lex Schrijver, and many other superstars of the area.

The chapter **"Computational Social Choice: The First Ten Years and Beyond"** by Aziz et al. [2] introduces the reader to a relatively new subarea of theoretical computer science. Computational social choice lies at the intersection between social choice theory (a subarea of economics and political science) and computer science. As a typical application, we want to mention *Dodgson's method* which is an electoral system that was proposed in 1876 by the British mathematician Charles Dodgson (who is much better known under the name Lewis Carroll, the author of Alice in Wonderland). On the positive side, *Dodgson's method* yields an electoral system that is very safe and extremely difficult to manipulate. On the negative side, Bartholdi et al. [3] have shown that even determining the winner of an election under Dodgson's method is an NP-hard problem. Other popular topics in computational social choice concern voting, coalition formation, matching markets, market equilibria, fair division, and cake cutting protocols. The chapter authors highlight several representative research areas in contemporary computational social choice, and also hint at future developments of the field.

The chapter **"Geometric Optimization Revisited"** by Agarwal et al. [1] deals with the area of computational geometry. Computational geometry is devoted to the study of algorithms that deal with points, lines, circles, triangles, polytopes, spheres, and other geometric objects. The area was triggered by computer graphics and computer-aided design, and more recently by geographic information systems, robotics, and computer vision. Typical problems studied in computational geometry concern motion planning, visibility problems, geometric location, geometric search, route planning, and mesh generation. Computational geometry forms an old and very mature part of algorithmics, and there are entire books available that summarize the main results [6,7]. The chapter concentrates on three concrete problems that have been very active in recent years: Geometric set cover problems; geometric independent set problems; and computing maps between point sets. These problems are typically hard to solve, and the chapter discusses various ways of working around this hardness.

The chapter **"Ten Reasons to Get Interested in Graph Drawing"** by Binucci et al. [4] introduces the reader to the area of graph drawing. (Note that there are ten authors who present ten reasons.) Graph drawing combines methods from geometry, graph theory, algorithmics, and information visualization to derive nice two-dimensional pictures of graphs that arise in application areas such as cartography, social network analysis and bioinformatics. A standard example is how to get a nice drawing of a subway network, with clearly readable names of subway stations and clearly recognizable changeover points. The chapter highlights ten concrete topics in graph drawing, four from theory (computational geometry; canonical orderings; the existential theory of the reals; SPQR-trees) and six from applications (information visualization; software engineering; model-based design; automated cartography; social sciences; molecular biology).

The chapter **"Sublinear-Time Algorithms for Approximating Graph Parameters"** by Ron [14] surveys the field of sublinear-time algorithms. For decades, researchers in algorithmics have been aiming for *linear-time* algorithms and considered them as the best-possible type of result that one can hope for. After all, it is hard to imagine that one could do better than linear-time: For any non-trivial problem, an algorithm should at least read and consider all of the input before making a qualified decision. However, (very) large data sets have become prevalent in a wide variety of settings, and it is natural to wonder what one can actually do in sublinear-time. With sublinear-time, one obviously can only read a miniscule fraction of the input. Typically, sublinear-time algorithms must use randomization and must give an answer which is (in some sense) approximate. Dana Ron discusses a variety of results on the sublinear approximation of certain graph parameters (average degree; higher moments of the degree distribution; number of connected components; vertex cover number; the weight of minimum spanning trees).

The chapter **"Dynamic Erdős-Rényi graphs"** by Mandjes et al. [11] deals with dynamic versions of the classical random graph model. Transportation networks, traffic networks, communication networks, and energy networks form the backbone of our modern society. Networks and graphs are used to model social, physical, chemical, biological, and technological phenomena. The existing literature predominantly focuses on static graphs: A random graph is drawn just once, and does not change over time. In many real-life situations, however, the network structure temporally evolves, with edges appearing and disappearing over time. The chapter discusses two dynamic versions of the classical Erdős-Rényi random graph. In the first version, the transition rates are governed by an external regime process, and in the second version the transition rates are periodically resampled. The chapter analyzes the corresponding moments, derives central limit theorems and investigates the large deviations asymptotics.

The chapter **"Wireless Network Algorithmics"** by Halldorsson and Wattenhofer [8] provides an introduction to algorithmic models for wireless networks. While wireless networks save us the costly process of introducing cables and make our lives easier, they also create new problems, as wireless transmission

may suffer from interference. To prevent interference, we could carefully schedule all transmissions so that concurrent transmissions are separated in space or time; however, scheduling a transmission at a different time might create new interferences at the newly chosen time slot. Another possibility would be to increase the transmission power in order to reduce interference; however, by increasing the transmission power, we may also create new interferences with other transmissions. The chapter presents the most popular mathematical models for wireless networks. It discusses a number of central results on link scheduling algorithms and on power control algorithms, it states future directions and poses several major open questions.

The chapter **"Green Computing Algorithmics"** by Pruhs [12] surveys the relatively new field of green computing, which perhaps could as well be called energy-aware computing. Nowadays one of the main problems in the design of new and faster VLSI chips is the generated heat, as it requires expensive (and noisy) cooling systems for computers. For instance, CEO Eric Schmidt from Google says: *"What matters most to the computer designers at Google is not speed, but power, low power, because data centers can consume as much electricity as a city."* Power consumption has led us to rethink and to redesign algorithmics from scratch, now with the minimization of energy as our top design constraint (instead of the somewhat outdated maximization of speed). The chapter explains why a theory of energy as a computational resource will look very different from the established theory of time or space as computational resources. Kirk Pruhs discusses various optimization problems under energy constraints, such as circuit routing in a network, scheduling on heterogeneous processors, and finding schedules that optimally trade off energy and performance.

The chapter **"Brain Computation: A Computer Science Perspective"** by Maass et al. [10] gives an introduction for computer scientists into brain computation. The human brain carries out tasks that are very demanding from a computational point of view, and apparently it is powered by a mere 20 Watts. The computational neuroscience pioneer David Marr has proposed a three-level approach to understanding brain computation: The behavioral level identifies the input-output behavior of the system, the algorithmic level analyzes the processes and representations used in the system, and the hardware level studies the biophysical elements and the molecular mechanisms used by the system to implement the algorithm. The chapter provides an overview of interactions between computer science and the study of computational aspects of the brain. The authors discuss the methodology of the computational study of the brain, while focusing on algorithms of the brain, on abstract and simplified models of brain systems, and on learning.

The chapter **"Rating Computer Science Via Chess"** by Regan [13] provides an introduction to (certain aspects of) computer chess. The chapter author Ken Regan is not only a well-known authority in theoretical computer science, but also a top notch chess player: In the 1970s he was one of the youngest chess masters since Bobby Fischer, and in 1980 he reached the international master (IM) level. In recent years Ken has also served as a member of the anti-cheating

committee of the World Chess Federation (FIDE), where he has been pivotal in writing the guidelines to prevent cheating in professional chess. By using his database with tens of thousands of top-level games, Ken has devised computer programs that can help to determine whether a player is playing more like a human or rather more like a computer. The chapter provides insight into the intersection between chess and computer science. Among other things, it discusses the complexity of endgames, the ways chess computers work, and it analyzes rating systems for the strength of chess players and chess programs.

The chapter **"Knowledge Harvesting: Achievements and Challenges"** by Weikum et al. [15] gives an overview on recent developments in the area of knowledge bases and knowledge harvesting. Knowledge bases are a technology used to store complex structured and unstructured information used by a computer system. Knowledge harvesting designs approaches for turning noisy Internet content into clearly structured information on entities and relations. Prominent examples of knowledge bases are the *Google* Knowledge Graph, the *True Knowledge Answer Engine* of Amazon, the semantic search engine *Facebook Graph Search*, and *Wolfram Alpha*. Prominent examples of knowledge harvesting are search engines like *Google* or *Bing*, semantic search, aggregating by entities, recommendations and data integration. The chapter surveys key principles of knowledge harvesting, summarizes the state of the art, discusses strategic challenges, and points out opportunities for future research.

References

1. Agarwal, P., Ezra, E., Fox, K.: Geometric optimization revisited. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 66–84. Springer, Cham (2018)
2. Aziz, H., Brandt, F., Elkind, E., Skowron, P.: Computational social choice: the first ten years and beyond. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 48–65. Springer, Cham (2018)
3. Bartholdi, J., Tovey, C.A., Trick, M.A.: Voting schemes for which it can be difficult to tell who won the election. Soc. Choice Welf. **6**, 157–165 (1989)
4. Binucci, C., et al.: Ten reasons to get interested in graph drawing. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 85–104. Springer, Cham (2018)
5. Cook, W.: Computing in combinatorial optimization. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 27–47. Springer, Cham (2018)
6. de Berg, M., Cheong, O., van Kreveld, M.J., Overmars, M.H.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77974-2
7. Edelsbrunner, H.: Algorithms in Combinatorial Geometry. Springer, Heidelberg (1987). https://doi.org/10.1007/978-3-642-61568-9
8. Halldorsson, M., Wattenhofer, R.: Wireless network algorithmics. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 141–160. Springer, Cham (2018)
9. Impagliazzo, R., Paturi, R.: The complexity of k-SAT. In: Proceedings of the 14th IEEE Conference on Computational Complexity, pp. 237–240 (1999)

10. Maass, W., Papadimitriou, C., Vempala, S., Legenstein, R.: Brain computation: a computer science perspective. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 184–199. Springer, Cham (2018)
11. Mandjes, M., Starreveld, N., Bekker, R., Spreij, P.: Dynamic Erdős-Rényi graphs. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 123–140. Springer, Cham (2018)
12. Pruhs, K.: Green computing algorithmics. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 161–183. Springer, Cham (2018)
13. Regan, K.: Rating computer science via chess. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 200–216. Springer, Cham (2018)
14. Ron, D.: Sublinear-time algorithms for approximating graph parameters. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 105–122. Springer, Cham (2018)
15. Weikum, G., Hoffart, J., Suchanek, F.: Knowledge harvesting: achievements and challenges. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 217–235. Springer, Cham (2018)
16. Williams, R.: Some estimated likelihoods for computational complexity. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 9–26. Springer, Cham (2018)