Preface

Parameterized complexity theory provides a framework for a refined analysis of hard algorithmic problems.

Classical complexity theory analyzes and classifies problems by the amount of a resource, usually time or space, that is required by algorithms solving them. It was a fundamental idea, going back to the work of Hartmanis and Stearns in the early 1960s, to measure the required amount of the resource as a function of the size of the input. This has led to a manageable variety of complexity classes and a clean-cut theory of intractability. However, measuring complexity only in terms of the input size means ignoring any structural information about the input instances in the resulting complexity theory. Sometimes, this makes problems appear harder than they typically are. Parameterized complexity theory takes a step backwards and measures complexity not only in terms of the input size, but in addition in terms of a *parameter*, which is a numerical value that may depend on the input in an arbitrary way. The main intention is to address complexity issues in situations where we know that the parameter is comparatively small.

Consider, for example, the problem of evaluating a database query. Its input has two parts, a database and the query. Observe that these two parts will usually differ in size quite significantly; the database will be much larger than the query. A natural parameter for a parameterized complexity analysis of this problem is the size of the query. As a more theoretically motivated example, consider approximation schemes for optimization problems. Their input consists of a problem instance and an error bound ϵ. A natural parameter is $1/\epsilon$. If we can accept an error of 5% in an approximation, we have a parameter value $1/\epsilon = 20$ for our approximation scheme. Typical parameters for many algorithmic problems on graphs are the tree width or the maximum degree of the input graph. Numerous other examples of naturally parameterized problems can be found in other application areas such as automated verification, artificial intelligence, or computational biology.

The central notion of parameterized complexity theory is *fixed-parameter tractability*. It relaxes the classical notion of tractability, polynomial time solv-

ability, by admitting algorithms whose "nonpolynomial behavior" is restricted by the parameter.

Of course, algorithms have always been analyzed and optimized in terms of many different input parameters, and no complexity theory was needed to do this. The main contribution of the theory is to provide a framework for establishing the *intractability* of certain problems. In the absence of techniques for actually proving lower bounds for natural problems, the main goal of such a theory is to classify problems into complexity classes by means of suitable reductions. Since the parameterized theory is two-dimensional, depending not only on the input size but also on the parameter, it is not surprising that it leads to a much larger variety of complexity classes and to more complicated reductions than the classical, one-dimensional complexity theory.

Besides providing a theory of intractability, parameterized complexity theory also provides a theory of fixed-parameter tractability that had significant impact on the design of algorithms. By consciously studying parameterized problems from different areas, researchers were able to devise new general algorithmic techniques for solving parameterized problems efficiently for small parameter values and to put existing algorithmic ideas into a larger context. Some of these general techniques are known as the method of bounded search trees, kernelization, color coding, and dynamic programming on tree decompositions.

An aspect of parameterized complexity theory that has gained importance more recently is its close connection with an area sometimes referred to as exact exponential worst-case complexity analysis. This area is concerned with exact algorithms[1] for hard algorithmic problems that are better than the trivial brute-force algorithms and corresponding (exponential) lower bounds for the running time of such algorithms. The role of the parameter in this context is to capture more precisely the main source of the (exponential) complexity of a problem. For example, the complexity of the satisfiability problem for formulas of propositional logic is better analyzed in terms of the number of variables of the input formula than in terms of its size.

Parameterized complexity theory is a fairly new branch of complexity theory. It was developed by Downey and Fellows in a series of ground breaking articles in the early 1990s. In these articles, Downey and Fellows defined the notion of fixed-parameter tractability, came up with suitable notions of reductions, defined the most important complexity classes, and proved a number of fundamental completeness results. Since then, numerous other researchers have contributed to the theory. Downey and Fellows' 1999 monograph [83] gives a fairly complete picture of the theory then. The development has not slowed down since then, quite to the contrary. However, we feel that the field has matured to a degree that it deserves a comprehensive state-of-the art introduction, which we hope to provide by this book.

[1] "Exact" as opposed to "approximation" algorithms.

Organization of This Book

In Chap. 1, we introduce the central notion of *fixed-parameter tractability*. We give various characterizations of fixed-parameter tractability. Furthermore, we explain one of the most basic algorithmic techniques for the design of fixed-parameter tractable algorithms (*fpt-algorithms* for short), which is known as the *method of bounded search trees*.

Intractability

Chapters 2–8 are devoted to the theory of fixed-parameter intractability. We start by defining an appropriate notion of reduction in Chap. 2. Then we turn to the question of what might be an analogue of the classical class NP in the world of parameterized complexity. In Chap. 3, we define and study the class W[P], which may be seen as such an analogue of NP. We develop a completeness theory for this class and establish its various connections with classical complexity theory.

Whereas natural problems in NP tend to be either in PTIME or NP-complete, there are many natural parameterized problems in W[P] that neither are in FPT nor are W[P]-complete. To classify such problems, we have to investigate the fine structure of the class W[P]. A skeleton for this fine structure can be obtained from *descriptive complexity theory*, which analyzes and classifies problems by the syntactic form of their definitions (in suitably formalized languages as provided by mathematical logic). It leads to a natural hierarchy of classes within W[P], the so-called W-*hierarchy*. Essentially, the levels of this hierarchy correspond to the number of alternations between universal and existential quantifiers in definitions of their complete problems. Many natural parameterized problems turn out to be complete for the first or second level of this hierarchy. The W-hierarchy is introduced in Chap. 5 and is further studied in Chap. 7. There is another hierarchy that extends beyond the boundaries of W[P], the so-called A-*hierarchy*. It may be seen as an analogue of the polynomial hierarchy in the world of parameterized complexity. The A-hierarchy is also introduced in Chap. 5 and is further studied in Chap. 8. The first levels of the A-hierarchy and the W-hierarchy coincide and are studied in detail in Chap. 6.

The necessary notions from mathematical logic and descriptive complexity theory are introduced in Chap. 4. Logic plays an important role in this book, not only in providing syntactical characterizations of the levels of the main hierarchies of intractable parameterized complexity classes, but also in *algorithmic metatheorems*, which state that all problems of a certain syntactic form are tractable. A well-known example for such a theorem is Courcelle's theorem, stating that all problems definable in monadic second-order logic are fixed-parameter tractable if parameterized by the tree width of the input structure.

Algorithmic Techniques

Chapters 9–14 are mostly devoted to advanced algorithmic techniques for designing fpt-algorithms. Our emphasis is always on the general techniques and not on optimizing the running times of algorithms for specific problems.

In Chap. 9 we study a technique known as *kernelization*. A kernelization algorithm reduces a given instance of a parameterized problem to a (presumably smaller) instance whose size is effectively bounded in terms of the parameter alone and does not depend on the size of the original instance. Thus kernelization is a form of preprocessing with an explicit performance guarantee. In the same chapter, we study the application of linear programming techniques to parameterized problems. So far, such techniques have only led to a few, albeit powerful, fixed-parameter tractability results.

In Chap. 10, we introduce the *automata-theoretic method*, which typically leads to fpt-algorithms, but not necessarily polynomial time algorithms. The automata-theoretic method plays a very important role in the design of algorithms for logic-based algorithmic problems, as they can be found, for example, in automated verification. A general theorem that can be obtained by automata-theoretic techniques states that the model-checking problem for monadic second-order logic on trees is fixed-parameter tractable. We also prove superexponential (actually, nonelementary) lower bounds for the running time of fpt-algorithms for this problem.

The following two chapters are devoted to algorithms on restricted classes of graphs and structures, specifically graphs of bounded tree width, planar graphs, and graphs with excluded minors. Based on the fixed-parameter tractability of the model-checking problem for monadic second-order logic on trees, which is proved in Chap. 10 by automata-theoretic means, we prove Courcelle's metatheorem mentioned above and give a number of applications of this theorem. We also (briefly) discuss the main algorithmic consequences of Robertson and Seymour's graph minor theory. Algorithms for planar graph problems have always received a lot of attention from researchers in parameterized complexity theory, and very refined techniques have been developed for designing fpt-algorithms on planar graphs. Some of these techniques can be generalized to larger classes of graphs, for example, classes of bounded local tree width. Chapter 12 is an introduction into this topic.

In Chap. 13, we study specific families of problems, *homomorphism problems* and *embedding problems* (also known as *subgraph isomorphism problems*). We obtain a complete classification of the complexity of certain restrictions of homomorphism problems, which essentially says that precisely the restrictions to instances of bounded tree width are tractable. Remarkably, for such problems fixed-parameter tractability and polynomial time solvability coincide. To prove the fixed-parameter tractability of restricted embedding problems, we introduce *color coding*, another general technique for designing fpt-algorithms.

Finally, in Chap. 14, we study the complexity of parameterized counting problems. After introducing counting versions of the most important parameterized complexity classes, we focus on the counting versions of embedding problems. We prove that counting paths or cycles in a graph is hard, even though by the results of the previous chapter the decision problems are fixed-parameter tractable. However, we show that the color coding technique of the previous chapter provides randomized approximation schemes for these counting problems.

Bounded Fixed-Parameter Tractability

The last two chapters of the book are devoted to a variant of fixed-parameter tractability that restricts the dependence of the running time of fpt-algorithms on the parameter. For instance, the dependence may be required to be singly exponential or subexponential. In this way, we obtain a whole range of different notions of *bounded fixed-parameter tractability*. The two most interesting of these notions are exponential fixed-parameter tractability and subexponential fixed-parameter tractability. We study these notions and the corresponding intractability theories in Chaps. 15 and 16, respectively. Both theories have interesting connections to classical complexity theory: The exponential theory is related to *limited nondeterminism* and, specifically, the class $\mathrm{NP}[\log^2 n]$ of problems that can be decided in polynomial time by a nondeterministic algorithm that only uses $\log^2 n$ nondeterministic bits. The subexponential theory provides a framework for exponential worst-case complexity analysis.

The dependencies between the chapters of this book are shown in Fig. 0.1. The two dashed arrows indicate very minor dependencies. In general, we have tried to make the chapters as self-contained as possible. A reader familiar with the basic notions of parameterized complexity theory should have no problem starting with any chapter of this book and only occasionally going back to the results of earlier chapters.

Throughout the book, we have provided exercises. Many of them are very simple, just giving readers an opportunity to confirm what they have just learned. Others are more challenging. We provide solution hints where we consider them necessary. At the end of each chapter we include a "Notes" section providing references for the results mentioned in the chapter and also pointers to further readings. At the end of some chapters, we also include a few open problems. It is not our intention to provide a comprehensive list of open problems in the area. We have just included a few problems that we find particularly interesting. We believe that most of them are quite difficult, some being long-standing open problems in the area.

Prerequisites

We assume familiarity with the basic notions of complexity theory, logic, and discrete mathematics. We provide an appendix with background material from

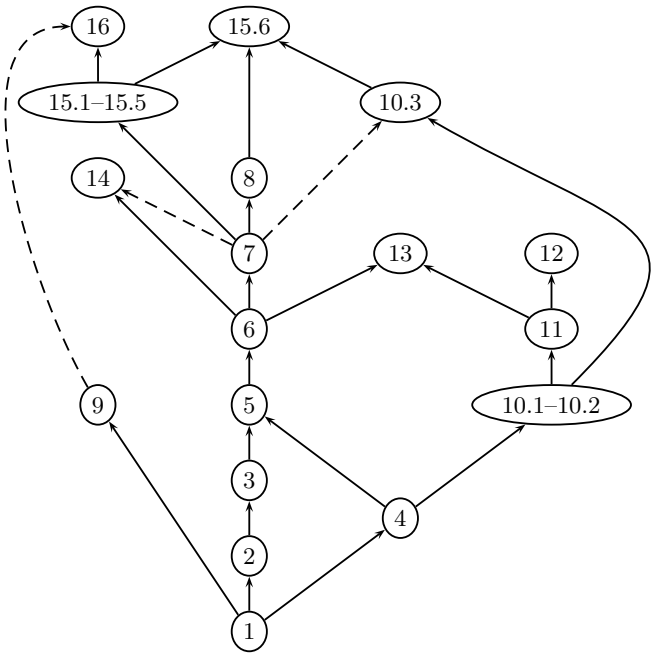

Fig. 0.1. Dependencies among the chapters and sections of this book

classical complexity theory that may serve as a reference for the reader. Knowledge of the appendix is no prerequisite for reading the book; the necessary definitions and notations are always introduced in the main text. A brief introduction to the relevant notions from logic and its connections with complexity theory is given in Chap. 4.

Acknowledgments

This book has greatly benefited from many discussions with our colleagues and students. We are especially indebted to Yijia Chen, who contributed to this book in many important ways, ranging from several nice proofs to such a prosaic thing as the preparation of the index. Numerous others helped with valuable comments, suggestions, and corrections. In particular, we would like to thank Dzifa Ametowobla, Argimiro Arratia, Fedor Fomin, Magdalena Grüber, Armin Hemmerling, Joachim Kneis, Stephan Kreutzer, Dániel Marx, Andrés Montoya, Moritz Müller, Cordula Ritter, Marcus Schaefer, Andreas Schlick, Nicole Schweikardt, Dimitrios Thilikos, Marc Thurley, and Mark Weyer.