Introduction

The *graph isomorphism problem* consists in deciding whether two given graphs are isomorphic, i.e. whether there is a bijective mapping (a permutation) from the nodes of one graph to the nodes of the second graph such that the edge connections are respected. The problem is of considerable practical, as well as theoretical, importance. Many isomorphism problems for other combinatorial structures can be considered as special cases of the graph isomorphism problem. However, until today, this problem is still unresolved in the sense that no efficient algorithm for it has yet been found. On the other hand, no NP-completeness proof for graph isomorphism, nor any other arguments for its intractability have been obtained either.

Under an efficient algorithm we understand one with running time bounded by some polynomial function in the input length (e.g., a polynomial in the number of vertices of the graphs). Given two graphs on n vertices, the naive algorithmic approach to solve graph isomorphism is to generate all $n!$ many permutations of the vertices and test them for being a solution, namely an isomorphism. This algorithm has complexity at least $n!$, and therefore, is certainly not efficient.

A more interesting algorithmic approach is to try to reduce the number of nodes to which a vertex can be mapped. This can be done by "labeling" the vertices of the two graphs in such a way that each vertex receives a label identifiying some property of the node in the graph. For instance, one can start out with the vertex's degree (the number of its neighbors) as a first component of a potential label, and then refine and distinguish equal labels further by other local properties of the vertex. If a labeling distinguishing all the vertices in each of the two graphs can be found, then it is trivial to find from it an isomorphism (if it exists) because vertices with equal labels should be mapped to each other. Such a labeling algorithm can be seen as a special way of identifying graphs efficiently with a "code" [112] (sometimes called "certificate" [102] or "complete sets of invariants" [54]) so that two graphs are isomorphic if and only if they have the same code. (A code for a graph could be obtained in this way ordering the vertex labels lexicographically and concatenating them in a string). It is clear that the existence of an efficient labeling algorithm (code producing algorithm) for graphs implies that the graph isomorphism problem can be solved efficiently, although the converse is not clear.

Although the labeling approach has not yet been totally successful in the sense that proposed methods work in all cases, they provide good algorithms in a variety of special cases. In other words, there exist classes of graphs on which certain algorithms can solve the isomorphism problem in polynomial time.[1] As a canonical example, we mention the class of *planar graphs*, for which efficient isomorphism tests exist [62, 63, 65]. Also, algorithms with moderately exponential time complexity (for the general problem) can be found this way [5].

Labeling algorithms have been shown to behave well *on average*. That is, for a random graph it turns out that a unique (or "unique enough") labeling can be found by some algorithm with high probability [10, 7] (see also [107, 135]). Therefore, under certain standard probability distributions for generating random graphs, algorithms can be devised for solving the isomorphism problem efficiently with high probability. (Against this it might be argued that two random graphs are non-isomorphic with very high probability. Therefore, an algorithm that answers always "non-isomorphic" would be very efficient—and very seldomly incorrect).

Taking in consideration that no efficient algorithm for the graph isomorphism problem is known, it is natural to ask whether such an algorithm exists at all, or whether, on the contrary, the problem is intractable. The theory of NP-completeness has been developed in order to deal with, or rather, to classify intractable problems. Already in the early works on NP-completeness the graph isomorphism problem was mentioned [33, 69, 43] and its importance was emphasized. The problem trivially lies in the class NP, and therefore the immediate question arises whether it is in P (i.e. efficiently solvable, as discussed before) or whether it is NP-complete (which would "solve" the problem as well, in some negative sense, providing strong evidence of its intractability). It has been shown [86] that if $\mathsf{P} \neq \mathsf{NP}$, then there exist problems that are neither in P nor NP-complete. Graph isomorphism might be a candidate for such an "intermediate" problem in NP. From the algorithmic point of view, a result saying that graph isomorphism is intermediate might not be satisfying because any problem outside of P is, by definition, not efficiently solvable (at least, in the worst-case), no matter whether it is NP-complete or not.

[1] Also, some special cases of graph isomorphism are known to be located in $\mathsf{NP} \cap \text{co-}\mathsf{NP}$ [58].

This book focusses on the last mentioned issues—rather than trying to devise other good algorithms or heuristics for graph isomorphism. We present several recent results that provide a better understanding of the relative position of the graph isomorphism problem in the class NP as well as in other complexity classes. All these results follow a line of research known as "structural complexity" that focuses on the structure of the complexity classes involved in the classification of computational problems, and in the relations among them, rather than on the design of specific algorithms. Our aim is to present such results in an, hopefully, accessible way to a broader audience. We hope that also people working in algorithms will become interested and will like to learn about such structural results. As it is said above, there is no direct consequence towards more efficient algorithms from the structural results, but nevertheless, we believe there is an indirect feed-back from such results to algorithm design. For example, we think it is an interesting observation that whenever a problem has turned out to be in some "low" structural class, like in $\mathsf{NP} \cap \text{co-}\mathsf{NP}$ or in $\oplus\mathsf{P}$, then it turns out that the problem has better-than-exponential-time (e.g. $O(n^{\log n})$) algorithms. Examples for this phenomenon can be seen in the computational problems of determining whether a number is prime [90], and the isomorphism problem for tournament graphs [96]. (Both NP problems are in $\oplus\mathsf{P}$, and the first is also in co-NP, see Section 4.1).

Besides presenting results that deal directly with graph isomorphism, we use this problem to illustrate important concepts in structural complexity, providing a look into the more general theory. This should give the reader a broader perspective to locate the specific graph isomorphism results; for example, we compare these particular results with the known ones for *satisfiability*, the best known NP-complete problem. Parts of this book may therefore have a more textbook-like character, in the sense of a textbook on structural complexity theory. We hope the reader will appreciate this.

The text is divided into four chapters and a preliminary section in which basic material and notation about graphs, isomorphisms, etc., is presented.

In the first chapter we consider the reducibility concept and the NP-completeness notion, the differences and similarities between different algorithmic tasks, like solving the decision version of a problem, the construction of a solution, or the counting version are studied. Further, it is shown how these concepts turn out to relate in the case of the graph isomorphism problem. The *graph automorphism problem* is also introduced in this chapter. It consists of determining whether a given graph has an automorphism other than the identity permutation. The close connections between this problem and the graph isomorphism problem are emphasized.

The second chapter deals with several useful structures and hierarchies of complexity theory that can be used to classify the complexity of the graph isomorphism problem: the polynomial-time hierarchy and the interactive proof systems. In this context several variants of probabilistic complexity classes are presented and studied in some depth. The notion of lowness that turns out to be very useful in classifying the graph isomorphism problem is introduced. The study of lowness—in connection with the graph isomorphism problem—will be continued in the further chapters.

In the third chapter, the possibility (or rather, impossibility) of reducing intractable problems to sets of low information content, such as "sparse" sets is studied. We show the close relations between sparse sets and sets that can be computed by polynomial size circuits, and we explore possible consequences of a reduction from the graph isomorphism problem to a sparse set.

The fourth chapter deals with the question of counting the number of solutions of a problem. We consider different counting classes and related structural complexity classes that can be used to capture graph isomorphism. In particular, it is proven that graph isomorphism is low for such classes, like PP, a result that combines different concepts in complexity theory and shows one further peculiarity of the problem.

A continuing theme in this book is the comparison between the graph isomorphism problem and **Sat**, as a canonical example for an NP-complete problem. Under different perspectives both problems turn out to be very different in nature, giving strong evidence that graph isomorphism is *not* NP-complete.