Course "Trees - the ubiquitous structure in computer science and mathematics ", JASS'08

Bounded treewidth

Stephan Holzer

April 28, 2008

Contents

1 Introduction and Motivation

On the one hand many of the problems we try to solve in practice are NP-complete and on the other hand we do not have the time to wait for years until the solution is found. Proving P=NP would help, but unfortunately we are far away from proving or disproving this major problem of theoretical computer science.

Therefore we can try to

- identify families of graphs $\mathfrak{G}$ which appear frequently in practice
- find efficient algorithms for restricted versions of NP-complete problems in the following sense: Let $\mathfrak{G}$ be a family of graphs like above and A a $NP-complete$ problem on an underlying graph G (like TSP, MIS, CLIQUE, ...). Now try to find an efficient algorithm for A when the underlying graphs are guaranteed to be in $\mathfrak{G}$.

An important example where we succeeded in finding efficient algorithms even for some PSPACE-complete problems are graphs of bounded treewidth which will be introduced in this survey.

2 History and Definitions

In the 1960s/70s researchers recognized that tree structures help making many difficult problems easy - to compute the result of a node, one only needs to know the result of its children. Fortunately many graph-classes allow algorithms to use a tree structure. First we will introduce series-parallel graphs as an important example of a class which allows the use of tree structures. We will see how to use them to compute the resistance of an electrical network by traversing a tree. Motivated by this we present a generalization of series-parallel graphs, namely graphs of bounded treewidth and provide examples of those graphs.

2.1 Series-parallel Graphs

Definition 2.1. *Series-parallel-graphs*
We define two-terminal s-p-graphs (G, a, b) inductively, where $G = (V, E)$ is an undirected graph, $a \in V$ a source terminal and $b \in V$ a sink terminal.

1. *The basic graph: (G, a, b) consisting of one edge between the two terminals: $V = \{a, b\}, E = \{(a, b)\}$*
2. *Serial composition: given two two-terminal s-p-graphs (G_1, a, c) and (G_2, c, b), take the disjoint union of G_1 and G_2, identify the two copies of node c, consider a and b to be the new terminals.*
3. *Parallel composition: given two two-terminal s-p-graphs (G_1, a, b) and (G_2, a, b), take the disjoint union of G_1 and G_2 , identify the two copies of node a, identify the two copies of node b, consider a and b to be the terminals of the resulting graph.*

Series-parallel graphs appear for example when computing the resistance of certain electrical networks when using Ohm's laws.

Example 2.2. *Recall Ohm's laws: the resistance R of a series composition of two resistors R_1 and R_2 is $R = R_1 + R_2$ and the parallel composition can be computed using the formula $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$.*

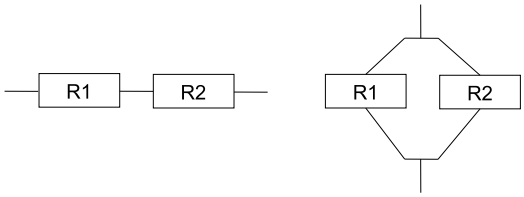

Figure 1: Serial and parallel compositions of resistors in electrical networks can be described by s-p-graphs

When computing the resistance of an electrical network N like in figure 2 one recognizes that we are computing along a tree-structure. In the tree the leave nodes correspond to the resistors and each internal node corresponds to a parallel or a serial composition of its children. We can compute the value of a node v by depth first search starting in v. Notice that when we finished computing the value of a node v we know the essential information of a subnetwork $N[v]$ of N represented by node v and can forget the internal structure of $N[v]$.

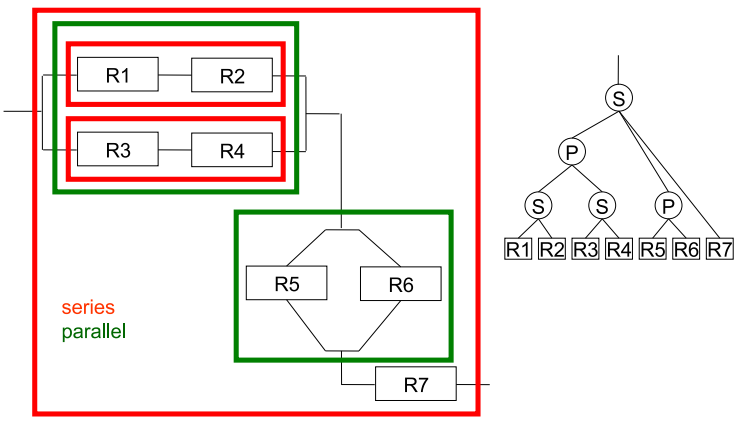

Figure 2: Computing resistance of a electrical network by using a tree-structure

2.2 Treewidth

We have noticed that tree-structures help to solve problems on graphs which are even more complex than trees. Therefore it is natural to define a family of graphs which should include as many graphs as possible and at the same time allows to design algorithms which can compute along a tree-structure. This family can be characterized by the notion of bounded tree-width which was introduced by Robertson and Seymour [RS83] in the 1980s! First we will define "tree-decomposition".

Definition 2.3. *Tree-decomposition*
Let $G = (V, E)$ be a graph. A tree-decomposition of G is a pair $(\{X_i | i \in I\}, T = (I, F))$ with $T = (I, F)$ a tree and $\{X_i | i \in I\}$ a family of subsets of V for each node $i \in I$ of T, such that

- $\bigcup_{i \in I} X_i = V$
- *if $(v, w) \in E$ is an edge, then there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$*
- *all nodes j on any i-k-path satisfy $X_i \cap X_k \subseteq X_j$*

In the following we will denominate the elements in V of G *vertices* and the elements in I of T *nodes*. A subset $X_i \subseteq V$ of the vertices of G corresponding to a node i of T will be called a *bag*.

Definition 2.4. *Treewidth*
The treewidth of a tree-decomposition $td = (\{X_i | i \in I\}, T = (I, F))$ is defined to be $treewidth(td) := \max_{i \in I} |X_i| - 1$. Now let G be a graph and $TD :=$ all tree-decompositions of G. Define $treewidth(G) := min_{td \in TD}(treewitdh(td))$

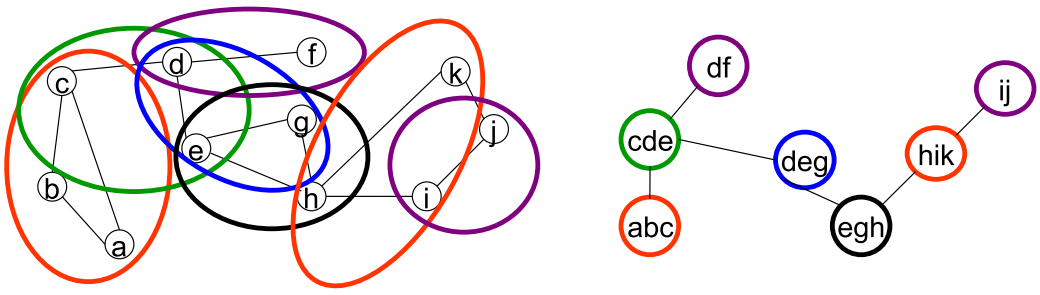

Figure 3: Example of one possible optimal tree-decomposition with treewidth 2

It is reasonable to subtract -1 like in the definition of treewitdh in order to ensure that the treewidth of a tree is 1.

2.3 Examples of Graphs of bounded Treewidth

The following table contains some common families of graphs as well as a brief description of each class and its treewidth. In *figure 4* the reader will find examples for each class which was not mentioned yet.

Class	Description	Treewidth
Trees	connected graph with no circles	1
Cycle	consists of a closed chain of vertices	2
Series-parallel graphs	see definition above	2
Outerplanar graphs	graph that has a planar embedding in the plane where all vertices lie on a fixed circle and the edges lie inside the disk of the circle	3
Halin graphs	tree in which all leaves are connected by a cycle and no node has degree 2	3
Pseudoforests	graph in which each connected component has at most one cycle	2
Cactus graphs	graph in which any two simple cycles have at most one vertex in common	2

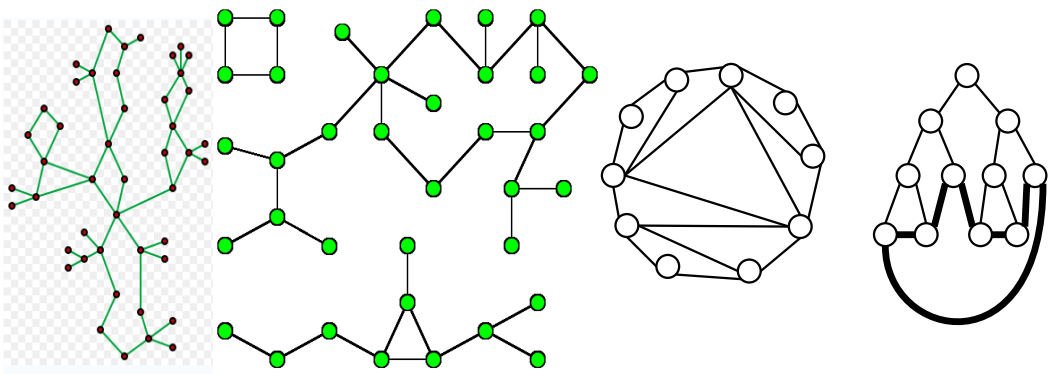

Figure 4: Examples of: cactus graph [wik], pseudoforest [wik], outerplanar graph, Halin graph

Example 2.5. *In this example we will determine the treewidth of a circle. In figure 5 we see a cycle consisting of five vertices and its tree-decomposition. Check that this tree-decomposition is legal and optimal. As no bag contains*

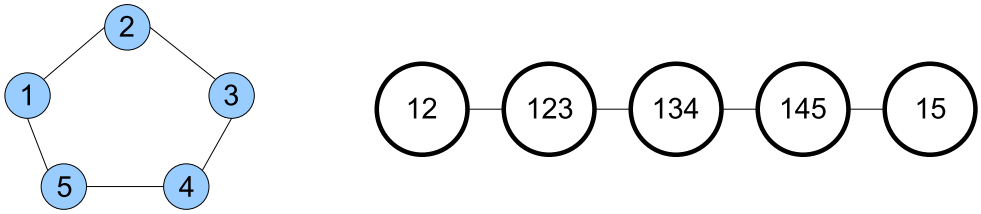

Figure 5: A cycle's optimal tree-decomposition

more than 3 vertices of the circle, the treewidth equals 2. For larger circles, the tree-decomposition can be extended in a canonical way.

All results for bounded treewidth apply to theses classes. Remark that planar graphs have no bounded tree width.

3 Applications

In this section we present the main ideas of some algorithms which use bounded treewidth to be more efficient than algorithms for general instances of unbounded treewidth. Bounded treewitdh is helpful in the context of Cholesky factorization, evolutionary theory, expert systems, VLSI layouts (via pathwidth - the tree T in the definition of tree-decomposition is replaced by a path P) and natural language processing. In an earlier talk of the JASS 2008 the Tutte-polynomial of graphs was introduced. In general this is a very hard problem but when the graph has bounded treewidth, it is computable in polynomial time [And98]. Furthermore, we will see a linear time algorithm for Independent Set restricted to bounded tree width graphs.

3.1 Cholesky Factorization on sparse Matrices

When we solve a system of linear equations we use the Gauß-elimination algorithm which can be interpreted as factorizing the corresponding matrix M into two triangular matrices. A special (50% faster) factorization can be achieved when M is a hermetian positive-definite matrix: the Cholesky factorization of a matrix M is a decomposition $M = L \cdot L^T$ with L a lower triangular matrix. Each step of this factorization is of the form

$$\begin{pmatrix} d & v^T \\ v & B \end{pmatrix} = \begin{pmatrix} \sqrt{d} & 0 \\ \frac{v}{\sqrt{d}} & I \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & \frac{B - v \cdot v^T}{\sqrt{d}} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{d} & \frac{v^T}{\sqrt{d}} \\ 0 & I \end{pmatrix}$$

Where v is an $(n-1)$-vector, B a $(n-1)\times(n-1)$-matrix, I the $(n-1)\times(n-1)$ identity matrix. The process is repeated with $B - v \cdot v^T$. To establish a connection to treewidth, let M be symmetric and sparse. Now consider the graph G with vertices $V = \{1, \cdots, n\}$ and edges $E = \{(i,j)|M_{i,j} \neq 0\}$. Removing one vertex of this graph and connecting all the neighboring vertices corresponds to one step of the factorization as described above.

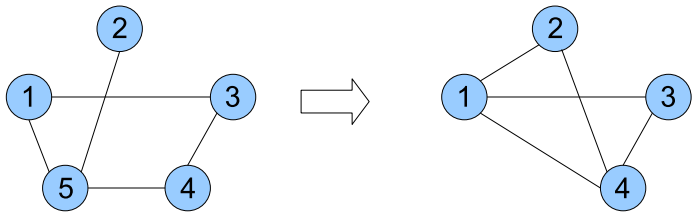

Figure 6: We remove vertex 5 and add a clique between vertices $\{1, 2, 4\}$

Our goal is to find an elimination order of the raws/columns such that all matrices $v \cdot v^T$ are small in the sense that there are many 0-rows/columns. This is possible as the matrix M is sparse. It turns out that bounding the size of the matrices $v \cdot v^T$ corresponds to bounding the treewidth of G.

3.2 Evolution Theory

In molecular biology we want to compute 'good' evolution trees, that is we are given as input a set of n species, a set of m characteristics and a $n \times m$-array consisting of the value that defines the relation of each specie to each characteristic. The goal is to compute a so called evolution tree for these species and their possibly extinct ancestors. One variant of this problem which we will consider is the PERFECT PHYLOGENY problem where

- the tree has n leaves and each leaf of the tree corresponds to one species from the input
- each character which appears in at least one species, is assigned to exactly one edge of T
- for each species i, the assignments of the characters to the edges of the path between the root and i correspond to the characters of i

We can turn this problem into an equivalent graph problem where the input is a graph $G = (V, E)$ and a legal coloring of V. Now we have to decide whether we can add edges to G in such a way that the resulting graph G' is chordal but has no edges between vertices of the same color, or not.

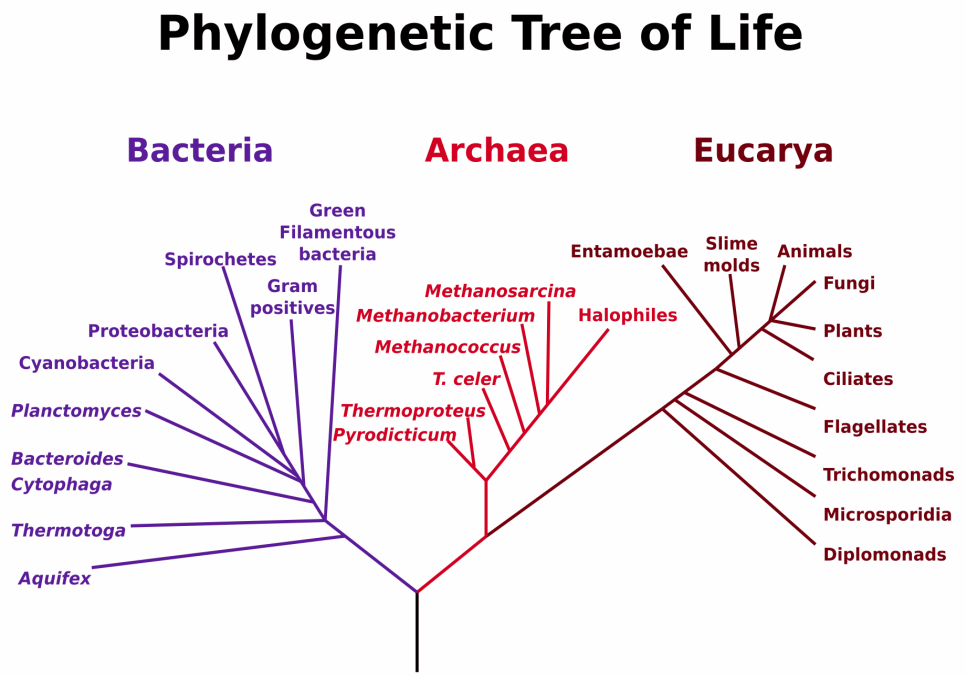

Figure 7: Example of a phylogenetic tree [wik]

Definition 3.1. *Chordal*
Let $G = (V, E)$ be an undirected graph. G is chordal if it contains no induced cycles $C_n, n \geq 4$.

This problem can be restated as: does there exist a tree-decomposition $(\{X_i | i \in I\}, T)$ of G such that for all $i \in I$: if $v, w \in X_i, v \neq w$, then v and w have different colors? Therefore a necessary condition is that $treewidth(G) <$ number of colors.

3.3 A linear time Algorithm for Maximum Independent Set

Maximum independent set is one of the most popular $NP - complete$ problems and defined as follows:

Definition 3.2. *Maximum independent set*
Let $G = (V, E)$ be a graph, find the maximum size of a set $W \subseteq V$ such that for all $v, w \in W : (v, w) \notin E$

Example 3.3.

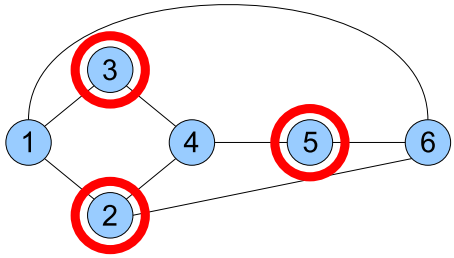

Figure 8: Example of a graph with the maximum independent set $\{2, 3, 5\}$

The situation changes when we know that G has treewidth k and we are given a tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ of G, where T is a binary tree (any tree-decomposition can be transformed into a binary one) and a root r of T. Then we can solve maximum independent set in linear time. This works as follows: for each $i \in I$ define

$$Y_i := \{v \in X_i | j = i \text{ or } j \text{ is a descendant of } i\}$$

If $v \in Y_i$ and $v \in X_j$ for some vertex $j \in I$ that is not a descendant of i then $v \in X_i$ by the definition of tree-decomposition. If $v \in Y_i$ and $v \in X_j$ adjacent to $w \in X_j$ with j a descendant of i then $v \in X_i$ or $w \in X_i$. Now we have turned the global problem of finding a MIS into a local (for every node $i \in I$ of the tree T) new problem: given a maximum independent set W of the subgraph $G[Y_i]$ induced by Y_i, extend W to a MIS of G. Here it is only important which vertices in X_i belong to W, not which vertices in $X_i - Y_i$ belong to W. Furthermore, the size $|W|$ of W plays a major role.

For $i \in I, Z \subseteq X_i$, define $is_i(Z)$ to be the maximum size of an independent set W in $G[Y_i]$ with $W \cap X_i = Z$. Take $is_i(Z) = -\infty$ if no such set exists. We are going to construct a linear time algorithm which computes all tables is_i in a bottom-up manner by using the following formula to compute all $2^{|X_i|}$ entries of the table is_i. Remark, that $|X_i|$ is bounded by a constant, namely the treewidth of G. For leaf nodes i we set:

$$is_i(Z) = \begin{cases} |Z| & \text{if } \forall v, w \in Z \quad : (v, w) \notin E \\ -\infty & \text{if } \exists v, w \in Z \quad : (v, w) \in E \end{cases}$$

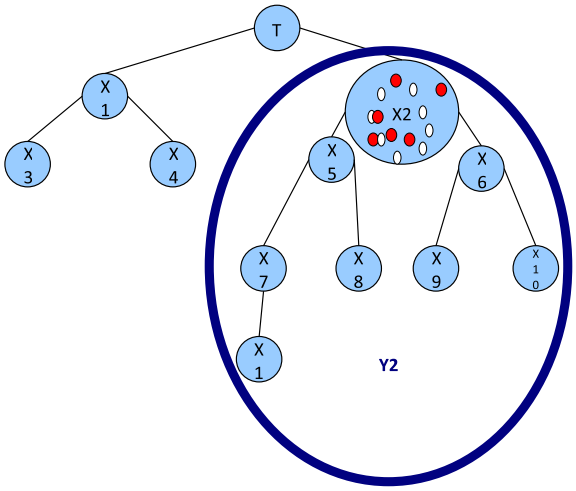

Figure 9: In the picture you see a tree-decomposition and the set Y_2 induced by the bag X_2. The red dots in X_2 represent one possible set Z of vertices, the vertices not in Z are colored white.

For internal nodes i (with two children j, k) we set:

$$is_i(Z) = \begin{cases} \max\{is_j(Z') + is_k(Z'') & \\ +|Z \cap (X_i - X_j - X_k)| & \\ -|Z \cap X_j \cap X_k| & \\ \text{where } Z \cap X_j = Z' \cap X_i & \\ \text{and } Z \cap X_k = Z'' \cap X_i\} & : \forall v, w \in Z : (v, w) \notin E \\ & \\ -\infty & : \exists v, w \in Z : (v, w) \in E \end{cases}$$

The solution will be $\max_{Z \subseteq X_r} is_r(Z)$ and the runtime is $O(2^{3k}n)$. Reconstructing the independent set is easy as well: one only needs to have a look on the tables is_i. However, remark that the constant 2^{3k} in the runtime grows exponentially in the treewidth.

3.4 (Extended) Monadic Second Order Formulas

There are many graph problems which can be expressed within (extended) monadic second order formulas, that is using the following language:

- constructions: logical operations $(\vee, \wedge, \neg, \Rightarrow)$

- quantification over vertices
- edges, sets of vertices, sets of edges (e.g. $\exists v \in V, \quad \forall e \in E, \quad \forall W \subseteq V, \quad \exists F \subseteq E$)
- membership tests ($v \in W, e \in E$)
- adjacency tests $(v, w) \in E, v$ is endpoint of e
- certain extensions

Theorem 3.4. *Let G be a graph of bounded treewidth. Graph problems on G expressible by monadic second order logic can be solved in linear time when given a tree-decomposition.*

Example 3.5. *We can express three-colorability of a graph in this language: There is a partition $W_1 \dot\cup W_2 \dot\cup W_3 = V$ of V into three colors which can be expressed by*

$$\exists W_1 \subseteq V \quad \exists W_2 \subseteq V \quad \exists W_3 \subseteq V \quad \forall v \in V : (v \in W_1 \vee v \in W_2 \vee v \in W_3)$$

To express that the two vertices v and w of each edge $(v, w) \in E$ should not have the same color we use

$$\forall v \in V \quad \forall w \in V : (v, w) \in E \Rightarrow$$
$$(\neg(v \in W_1 \wedge w \in W_1) \wedge \neg(v \in W_2 \wedge w \in W_2) \wedge \neg(v \in W_3 \wedge w \in W_3))$$

By concatenating the two expressions, we obtain the desired description of three-colorability.

4 Finding Tree-decompositions

In the last sections we saw that treewidth helps to speed up algorithms for "easy" instances of hard problems. However, in those algorithms, it is always necessary to know a (optimal) tree-decomposition of the given graph. Now there arises a new problem: it is NP-complete to decide whether a (general) graph $G = (V, E)$ has treewidth $\leq k$ for some integer k? Fortunately there exist algorithms for constant k with polynomial runtime. At the same time we can compute an approximation of the tree-decomposition.

Theorem 4.1. *For constant treewidth $k = 1, 2, 3, 4$, those graphs can be recognized in linear time*

Theorem 4.2. *Let k be a constant, then there exists an $O(n \cdot log(n))$ algorithm, that given a graph $G = (V, E)$, either outputs*

- *$treewidth(G)$ is larger than k*
- *or a tree-decomposition of G with treewidth at most $3k + 2$*

Class	Treewidth
Bounded degree	N [35]
Trees/Forests	C
Series-parallel graphs	C
Outerplanar graphs	C
Halin graphs	C [143]
k-Outerplanar graphs	C [20]
Planar graphs	O
Chordal graphs	P (1)
Starlike chordal graphs	P (1)
k-Starlike chordal graphs	P (1)
Co-chordal graphs	P [85]
Split graphs	P (1)
Bipartite graphs	N
Permutation graphs	P [34]
Circular permutation graphs	P [34]
Cocomparability graphs	N [6, 72]
Cographs	P [36]
Chordal bipartite graphs	P [86]
Interval graphs	P (2)
Circular arc graphs	P [135]
Circle graphs	P [83]

Figure 10: Table from [Bod94] showing the complexity of computing the treewidth of a graph according to its classification. P =polynomial time solvable. C =constant and therefore linear time solvable, N =NP-complete. O =open problem.

References

[And98] A. Andrzejak. An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics*, 190(1-3):39–54, 1998.

[Bod] H.L. Bodlaender. Treewidth: Characterizations, applications, and computations. *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG*, 6:1–14.

[Bod94] H.L. Bodlaender. A tourist guide through treewidth. *Developments in Theoretical Computer Science*, 1994.

[Die00] R. Diestel. *Graphentheorie*. Springer, 2000.

[RS83] N. ROBERTSON and PD SEYMOUR. Graph minors. I: Excluding a forest. *Journal of combinatorial theory. Series B*, 35(1):39–61, 1983.

[wik] http://en.wikipedia.org.