Treewidth: Algorithmic techniques and results*

Hans L. Bodlaender†

Abstract

This paper gives an overview of several results and techniques for graphs algorithms that compute the treewidth of a graph or that solve otherwise intractable problems when restricted graphs with bounded treewidth more efficiently. Also, several results on graph minors are reviewed.

1 Introduction

The notion of treewidth is playing a central role in many recent investigations in algorithmic graph theory. There are several reasons for the interest in this, at first sight perhaps somewhat unnatural notion. One of these reasons is the central role that the notion plays in the theory on graph minors by Robertson and Seymour (see Section 5); another reason is that many problems that are otherwise intractable become polynomial time solvable when restricted to graphs of bounded treewidth (see Section 4).

There are several 'real world' applications of the notion of treewidth, amongst others in expert systems [91], telecommunication network design ([46]), VLSI-design, Choleski factorization, natural language processing [89] (see e.g. [21] for a brief overview.) An interesting recent application has been found by Thorup [127]. He shows that for many well known programming languages (like C, Pascal, Modula-2), the control-flow graph of goto-free programs has treewidth bounded by a small constant (e.g., 3 for Pascal, 6 for C). Thus, certain optimization problems arising in compiling can be solved using techniques relying on small treewidth.

2 Definitions

The notion of treewidth was introduced by Robertson and Seymour in their work on graph minors [101].

*This research was partially supported by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

†Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Email: hansb@cs.ruu.nl.

Definition. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{X}, T)$ with $T = (I, F)$ a tree, and $\mathcal{X} = \{X_i \mid i \in I\}$ a family of subsets of V, one for each node of T, such that

- $\bigcup_{i \in I} X_i = V$,
- for all edges $\{v, w\} \in E$ there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$, and
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition $((I, F), \{X_i \mid i \in I\})$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

There are several equivalent notions to treewidth (for an overview, also of classes of graphs that have a uniform upper bound on the treewidth, see [25]); amongst others, graphs of treewidth at most k are also known as *partial k-trees.* A notion related to treewidth is *pathwidth*, defined first in [99]. A tree decomposition $(\mathcal{X}, T)$ is a path decomposition if T is a path; the pathwidth of a graph G is the minimum width over all path decompositions of G. A survey giving relations to notions of graph searching has been written by Bienstock [14].

Another notion that is related to treewidth and that might be more suitable in some cases for implementation purposes is *branchwidth* [115].

A tree decomposition can easily be converted (in linear time) in a *nice* tree decomposition of the same width (and with a linear size of T): here the tree T is rooted and binary, and nodes are of four types:

- Leaf nodes i are leaves of T and have $|X_i| = 1$.
- Introduce nodes i have one child j with $X_i = X_j \cup \{v\}$ for some vertex $v \in V$.
- Forget nodes i have one child j with $X_i = X_j - \{v\}$ for some vertex $v \in V$.
- Join nodes i have two children j with $X_i = X_{j_1} = X_{j_2}$.

Using nice tree decompositions instead of normal ones does in general not give additional algorithmic possibilities, but it considerably eases the design of algorithms, and one can also expect in several cases to have better constant factors in the running times of algorithms that use nice instead of normal tree decompositions.

3 Determining treewidth

In this section we review a number of results on the problem, to determine the treewidth of a given graph.

The problem to determine, when given a graph G and an integer k, whether the treewidth of G is at most k is NP-complete [5], even for graphs of maximum degree at most 9 [36], bipartite graphs, or cocomparability graphs. For several special graph classes, there exist polynomial time algorithms to determine the treewidth of graphs in the class, e.g., for chordal graphs, permutation graphs [33], circular arc graphs [123], circle graphs [85], and distance hereditary graphs [40]. See also [34, 60, 71, 76, 77, 87]. One of the most interesting open problems here is the complexity of treewidth when restricted to planar graphs. As branchwidth can be solved in polynomial time on planar graphs [122], and branchwidth differs at most a factor 1.5 from treewidth, we have a polynomial time approximation algorithm for treewidth on planar graphs with performance ratio 1.5. For arbitrary graphs, there is a polynomial time approximation algorithm for treewidth with performance ration $O(\log n)$ [30]; it is an interesting (but probably hard) open problem whether treewidth can be approximated with constant performance ratio.

A well studied case is when the parameter k is a fixed constant. We distinguish here results for two versions of the problem: the decision problem, where we only must decide whether the treewidth of the input graph is at most k, and the construction problem, where also a tree decomposition of width at most k must be given, when existing.

The first polynomial time algorithm for the (construction and decision) problem used $O(n^{k+2})$ time and was found by Arnborg, Corneil, and Proskurowski [5]. Using deep results on graph minors, Robertson and Seymour then gave a non-constructive proof of the existence of a decision algorithm that uses $O(n^2)$ time [115]. This algorithm has the following structure. First, in $O(n^2)$ time, we can find a tree decomposition of the input graph G with width at most $4k+3$, or decide that the treewidth of G is at most k. (To be precise, Robertson and Seymour use branchwidth to give a similar result; the technical difference is not important.) Then, this tree decomposition is used to check in linear time whether G contains an element of the obstruction set of graphs of treewidth at most k (see section 5.)

The first step of this algorithm was improved by Matousek and Thomas [94], who gave a faster randomized algorithm, by Lagergren [90], who gave a parallel algorithm using $O(log^3 n)$ time and $O(n)$ processors on a CRCW PRAM, or a sequential $O(nlog^2 n)$ time algorithm, and Reed, who gave an algorithm running in $O(n \log n)$ time. Each of these algorithms either determines that the treewidth of input graph is more than G or finds a tree decomposition of width at most $f(k)$ for some linear function f. (See [42] for a simple linear time algorithm for the pathwidth variant of this problem.) In [32, 86], Bodlaender and Kloks address the second step of the algorithm of Robertson and Seymour: they give an algorithm for the second step that solves the construction problem in linear time (i.e., provided a tree decomposition of bounded but perhaps not optimal width has been found). Using the algorithm from [32], in [24] for each fixed k, a *linear* *time* algorithm is given for both the decision and construction problem. Each of the algorithms mentioned in this paragraph has a hidden constant factor that is at least exponential in k — in all cases, the factors seem too large for practical purposes, but little experimental work has been done so far.

An interesting different approach was taken by Arnborg et al. [6]. They use graph reduction to obtain algorithms that use linear time, but more than linear memory. More on graph reduction can be found in Section 4.

Work has also been done on parallel algorithms for the fixed parameter case of the treewidth problem. Older algorithms by Bodlaender [17] and Chandrasekharan and Hedetniemi [43] need large numbers of processors. The first algorithm with work (product of time and number of processors) only a polylogarithmic factor more than linear was the algorithm by Lagergren [90], discussed above. This result was improved by Bodlaender and Hagerup [31], who, combining parallizations of the sequential algorithms of [24] and [6] with new techniques, obtained the following results. On an EREW PRAM, the construction problem can be solved in $O(log^2 n)$ time; the decision problem can be solved in $O(\log n \log^* n)$ time on a EREW PRAM and $O(\log n)$ time on a CRCW PRAM. Each of the algorithm has optimal speedup, i.e., the product of time and number of processors is linear.

In the case that k is 2, 3, or 4, better algorithms have been found. Practically efficient linear time algorithm exist, based on graph reduction [8, 94, 119]. The parallel algorithms for $k = 2, 3$ by [73] were improved by the results in [31], mentioned above. For $k = 2$, a parallel algorithm for the construction problem that uses $O(\log n \log^* n)$ time on a EREW PRAM and $O(\log n)$ time on a CRCW PRAM and has optimal speedup has been found by de Fluiter and Bodlaender [54, 55].

See also [37] for a closely related problem.

4 Finding algorithms for problems on graphs of small treewidth

For large numbers of graph problems, it has been shown that they are solvable in linear time, polynomial time, or become a member of NC, when the inputs are restricted to graphs of treewidth at most k for some constant k. Underlying many of these results, there are a few common techniques. In this section, these techniques are reviewed.

A technique that is very often applicable for solving problems on graphs of bounded treewidth is the one, discussed below. It can be characterized as: 'computing tables of characterizations of partial solutions' for each node $i \in I$ in a tree decomposition of bounded width – in bottom-up order. The technique first appeared (in 1992) in the context of graphs of bounded treewidth in a paper by

Arnborg and Proskurowski [9]; another paper founding this technique was Bern et al. [13].

The algorithm has the following structure (k is assumed the assumed constant upper bound on the treewidth of input graph $G = (V, E)$):

- Find a tree decomposition of G of width at most k. (This can be done in linear time, as discussed above.)

- Transform it into a nice tree decomposition, say $(\{X_i \mid i \in I\}, T = (I, F))$ of width at most k, $|I| = O(|V|)$, r the root of T.

- Compute for each node $i \in I$ a certain table. To compute a table for a node i, one only uses tables already computed for the children of i, the type of node i (leaf, forget, introduce, or join), and the information about G, restricted to X_i. Thus, these tables are computed in bottom-up order.

- The answer to the problem can be found by inspecting the table of the root r.

- Construction versions of problems usually need another phase, where tables are used again to construct a solution (when one exists). We will not go into detail for this step.

To describe what type of tables are needed, we first introduce some additional notions.

A terminal graph is a triple $H = (V, E, X)$, where (V, E) is a graph with vertex set V and edge set E, and X is an ordered subset of the vertices in V, called the *terminals* of G. A terminal graph with l terminals is also called an l-terminal graph. The operation $\oplus$ is defined on pairs of terminal graphs with the same number l of terminals: $H \oplus H'$ is obtained by taking the disjoint union of H and H' and then identifying the ith terminal of G with the ith terminal of H' for all i, $1 \leq i \leq l$. A terminal graph H is a terminal subgraph of a graph G, iff there exists a terminal graph H' with $G = H \oplus H'$.

To every node i in a nice tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ we associate the terminal graph $G_i = (V_i, E_i, X_i)$, where V_i is the set $\{v \mid v \in X_j$ and $j = i$ or j is a descendant of i in $T\}$; $E_i = E[V_i] = \{\{v, w\} \in E \mid v, w \in V_i\}$, or, in other words: the subgraph induced by vertices in the sets of the node j and all nodes below j in T, with X_i as the set of terminals. (The ordering of X_i is not important.)

At this point, we are ready to describe the 'build tables of partial solutions technique' more precisely. Suppose we want to solve a certain problem X, where for the moment we assume that X is a graph property. The algorithm design follows the following steps.

1. Define a notion of *solution*. For instance, if X is the Hamiltonian circuit problem, then a solution for a graph G is an actual Hamiltonian circuit in G.

2. Define a notion of *partial solution*. A partial solution is an object that can be associated with a terminal graph. When this terminal graph H is a terminal subgraph of a graph G, then the partial solution should describe possible behavior of a solution on G, when we 'only look to what happens on H'. For instance, a partial solution for Graph Coloring is a coloring of the vertices of the terminal graph, a partial solution for Hamiltonian circuit is a set of paths between the terminals in the terminal graph, disjoint except possibly for their endpoints and covering all vertices in the terminal graph.

3. Define a notion of *extension of partial solution*. We must specify what it means that a solution is an extension of a partial solution. This is usually very natural, for instance, for graph coloring, a solution for $G = (V, E)$, i.e., a coloring f of G, is an extension of a partial solution (coloring g) for terminal subgraph $H = (V', E', X)$ iff g is the restriction of f to W.

4. Define a notion of *characteristic* of a partial solution. It is meant to describe 'what is needed to know about the partial solution to see whether it can be extended to a solution', i.e., if two partial solutions have the same characteristic, then one can be extended to a solution if and only if the other can be extended to a solution. See below for examples.

5. A *full set of characteristics* for a terminal graph G is the set of all characteristics of partial solutions on G. The full set of characteristics of a graph G_i for a node i in a nice tree decomposition is also called the full set for i. Show for each of the four types of nodes (leaf, introduce, forget, join), that for node i, a full set for i can be computed efficiently (in constant or polynomial time), given the full sets for all children of i, assuming that there are at most $k + 1$ terminals in each of the involved terminal graphs.

6. Show that the problem can be decided efficiently (in constant or polynomial time), given a full set for the root node r of the nice tree decomposition.

More formally: we have relations sol_X, $psol_X$, ex_X, and a function ch_X, capturing respectively the notions 'solution', 'partial solution', 'extension', and 'characteristic'. sol_X is a relation with two arguments (G, s), G a graph and s a 'solution string', such that for all graphs G: $X(G) \Leftrightarrow \exists s : sol_X(G, s)$. (This is comparable to a definition of NP.) $psol_X$ is a relation with two arguments (H, s), H a terminal graph, and s a 'partial solution string'. ex_X is a relation with four arguments (G, s, H, s'), G a graph, s a solution string, H a terminal graph, s' a terminal solution string. The following must hold, for all G, s, H, s'

$$ex_X(G, s, H, s') \rightarrow \exists H' : G = H \oplus H' \wedge sol_X(G, s) \wedge sol_X(H, s')$$

Also, the following must hold, expressing that every solution has a partial solution on any terminal subgraph:

$$\forall G, s, H, H' : (sol_X(G, s) \wedge G = H \oplus H') \rightarrow \exists s' : psol_X(H, s') \wedge ex_X(G, s, H, s')$$

ch_X is a function on pairs (H, s), H a terminal subgraph and s a partial solution string, defined when $psol_X(H, s)$ is true. It must fulfill, for all terminal graphs H, H', H'', partial solution strings s, s':

$$ch_X(H, s) = ch_X(H', s') \Rightarrow (\exists s'' : ex_X(H \oplus H'', s'', H, s)) \Leftrightarrow (\exists s''' : ex_X(H' \oplus H'', s''', H', s')) \quad (1)$$

The full set of characteristics of terminal graph H is $full_X(H) = \{ch_X(H, s) \mid psol_X(H, s)\}$.

As example, we look to the problem to decide whether a graph $G = (V, E)$ has chromatic number at most three, i.e., is there an $f : V \rightarrow \{1, 2, 3\}$, such that $\forall \{v, w\} \in E : f(v) \neq f(w)$?

The notions 'solutions', 'partial solution', and 'extension' are obvious, and discussed above. The characteristic of a partial solution $f : W \rightarrow \{1, 2, 3\}$ of terminal graph $H = (W, F, X)$ is the restriction of f to X, $f|_X$. The following lemma shows the correctness of this notion of characteristic.

Lemma 4.1 *Let $H = (V, E, (v_1, \ldots, v_l))$, $H' = (V', E', (v'_1, \ldots, v'_l))$ be l-terminal graphs. Let c, c' be 3-colorings of the vertices of H and H', such that for $1 \leq i \leq l$, $c(v_i) = c'(v'_i)$. Then for any l-terminal graph H'', there exists a 3-coloring of $H \oplus H''$ that extends c, if and only if there exists a 3-coloring of $H' \oplus H''$ that extends c''.*

The proof of the lemma is easy: color the new vertices in H'' in both cases is the same way. An important element in the proof is the following observation: there are no edges between a vertex that belongs to H but not to X, and a vertex that belongs to H' but not to X in the graph $H \oplus H'$.

Next, notice that for any node i in the nice tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of $G = (V, E)$, G is of the form $G = G_i \oplus H$ for some terminal graph H. This follows from the definition of tree decomposition: the only vertices in V_i, adjacent to vertices in $V - V_i$ are those that belong to X_i.

What this all amounts to is that the full set of characteristics of G_i is actually all one needs to know about the terminal subgraph G_i when solving the 3-coloring problem. Additionally, as for each i, $|X_i| \leq k + 1$, if G is a graph of treewidth at most k, each full set contains at most 3^{k+1} elements, which is constant when k is a constant. It is not hard to show that full sets can be computed in constant time, given the full sets of the children of a node. This has to be shown for each of the four cases: leaf node, introduce node, forget node, join node. We only look at the case of introduce node here.

Lemma 4.2 *Let $\chi 3$ be the 3-coloring problem, $(\{X_i \mid i \in I\}, T = (I, F))$ a nice tree decomposition of $G = (V, E)$, i an introduce node with child j, and $X_i = X_j \cup \{v\}$. A function $f : X_i \to \{1, 2, 3\}$ belongs to $full_{\chi 3}(i)$, if and only if $f|_{X_j} \in full_{\chi 3}(j)$ and for all $w \in X_j$, if $\{v, w\} \in E$, then $f(v) \neq f(w)$.*

The proof relies on the fact that v can only be adjacent in G_i to vertices in X_j — this follows from the definition of tree decomposition. The lemma shows that we can compute full sets for introduce nodes, given a full set of the child, and only very local information. Similar lemmas exist for the other types of nodes.

Computing full sets in bottom up order, we finally have a full set for the root node. Now, as G is 3-colorable, if and only if the full set for the root node is non-empty, we can directly decide the problem.

When we used a tree decomposition of width, bounded by a constant, each computation of a full set took time, only exponential in that constant, hence the entire algorithm uses linear time.

Similar approaches work also for many other problems. In other cases, notions of partial solution, extension, and especially characteristic are less obvious or hard to find. For instance, look at the problem to decide, whether on a graph $G = (V, E)$, given with a number of pairs (v_i, w_i), $1 \leq i \leq r$ of vertices, there are paths from each v_i to w_i that are mutually disjoint. Clearly, a solution here is the desired set of paths. A partial solution for a terminal subgraph H is a collection of disjoint paths, some of which end in a terminal, such that certain properties hold: e.g., if v_i and w_i both belong to H, there either must be a path between v_i and w_i in the subgraph, or both v_i and w_i must have a path ending in a terminal (there then must be another path joining these terminals in the solution). Note there can also be paths between terminals. The characteristic of such a partial solution then describes which terminals are joined with which vertices from pairs; one can actually show there are at most a constant number of possibilities when $|X_i|$ is bounded by a constant, and that for each type of node, a full set can be computed in constant time. See [121].

When designing these types of algorithms, the most important step is the right choice of characteristic. First, it should fulfill property (1). Secondly, one should aim for characteristics, such that full sets of l-terminal graph (or l-terminal graphs of bounded treewidth) have bounded size, i.e., size only depending on l. Experience shows, that once the right choices for solution, partial solution, extension, and characteristic are made, the design of the algorithm (i.e., procedures how to compute full sets for the four types of nodes, and deciding the property given the full set of the root) usually succeeds, although it often is a lot of detailed work.

A similar technique works for optimization problems. We omit the more formal framework here, and give only a sketch. For characteristics, we take pairs (s, r), with r a member of a totally ordered set, usually an integer or real number — the value of the partial solution. If we have partial solutions with characteristics (s, r_1) and (s, r_2), then we put only that characteristic of these in the full set with the best value r_1 or r_2.

For instance, suppose we want to solve the problem to find a mapping f of the vertices of a graph $G = (V, E)$ to colors 1, 2, 3, such that the number of edges $\{v, w\} \in E$ with $f(v) = f(w)$ is as small as possible. A partial solution for a terminal graph $H = (W, F, X)$ is a coloring f of the vertices in W; the characteristic of this partial solution is the pair $(f|_X, r)$, where $f|_X$ is the restriction of f to X, and r is the number of edges $\{v, w\} \in F$ with $f(v) = f(w)$. For each possible function $g : X \to \{1, 2, 3\}$, we have at most one pair (g, r) in the full set of H, namely the pair with the smallest possible value of r. Again, we can compute the full set for a node when given the full sets for its children in a tree decomposition in constant time, giving a linear time algorithm. (For similar algorithms, see e.g., [80].)

Actually, results exploiting analogues to Myhill-Nerode theory (for finite state automata) can be used to show existence of an algorithm at an earlier stage of the design process, when dealing with certain types of decision problems.

Let P be a graph property. Define the relation $\sim_{P,k}$ on k-terminal graphs as follows:

$$G \sim_{P,k} H \Leftrightarrow (\forall K : P(G \oplus K) \leftrightarrow P(H \oplus K))$$

We say that P is of *finite index*, when for every k, the equivalence relation $\sim_{P,k}$ has a finite number of equivalence classes.

One can show that every finite index problem can be solved in linear time on graphs of bounded treewidth (see e.g., [2]). Now, as soon as we have characteristics which need $O(1)$ bits to describe, we know that the problem is finite state: if k-terminal graphs G and H have the same full set, then $G \sim_{P,k} H$, and there are only a constant number of different possible full sets.

Graph reduction Another interesting algorithmic method is based on graph reduction. Here, we observe that when $H \sim_{P,k} H'$, then when we have a graph of the type $H \oplus K$, we can replace it by $H' \oplus K$ and not change the answer of the problem. When H' is smaller than H, we have reduced the problem to an equivalent one of equal size. If P is of finite index, one can show that there exists a set of such 'safe' graph reduction rules for P, that can be used for a linear time algorithm of the following form: repeatedly apply a reduction rule to G. When no rule can be applied, we have a graph of size at most some constant, or for which P does not hold. This method was introduced to the setting of treewidth in [6]. More on graph reduction can be found in [26, 54].

Monadic second order logic An interesting general framework to quickly establish that a problem can be solved in linear time on graphs of bounded treewidth has been established by Courcelle [49, 48, 47, 51], and extended by Borie et al. [39], Arnborg et al. [7], and Courcelle and Mosbah [53]. Courcelle results states that each problem that can be stated in *monadic second order logic* can be solved in linear time on graphs of bounded treewidth. Monadic second order logic is a language to describe graph properties, using the following constructions: quantifications over vertices, edges, sets of vertices, sets of edges, ($\exists v \in V$, $\forall F \subseteq E$, ...); membership tests ($v \in W$, $e \in F$), adjacency tests (v is endpoint of e), and logic operations ($\vee$, $\neg$, ...). Extensions allow e.g., to optimize over the size of a free set variable. For instance, the problem whether a graph has a partition of its vertices in triangles (i.e., we want to partition V into $V_1, \ldots, V_r$, such that each V_i has three vertices and induces a triangle in G) can be expressed as:

$$\exists F \subseteq E\colon \forall v \in V\colon \exists w \in V\colon \exists x \in V\colon \{v,w\} \in F \wedge \{v,x\} \in F \wedge \{w,x\} \in F \wedge \neg\, (\exists y \in V : y \neq w \wedge y \neq x \wedge \{v,y\} \in F\}).$$

(To be precise, instead of $\{v,w\} \in F$, we should write: $\exists e : e \in F \wedge v \in e \wedge w \in e$.) The maximum independent set problem can be formulated as:

$$\max_{W \subseteq V} |W| : \forall v : \forall w : (v \in W \wedge w \in W) \rightarrow \neg(\{v,w\} \in E)$$

Especially the paper by Borie et al. [39] is very helpfull to see what kind of constructions can be used to express problems in (extensions of) monadic second order logic. Problems in monadic second order logic are finite index.

An interesting question is whether language constructions can be added to monadic second order logic, such that its expressive power becomes sufficient to describe all problems that are finite index. See e.g., [50, 52, 81].

Additional remarks Some problems whose decision versions are not in NP can also be solved in linear time on graphs of bounded treewidth. See e.g. [4, 3, 19].

For more algorithms that exploit the small treewidth of graphs, see also (amongst others) [10, 18, 38, 44, 61, 74, 79, 92, 93, 94, 125, 126, 129, 130].

Dynamic algorithms for graphs of bounded treewidth have been considered amongst others in [22, 45, 70, 78]. Parametric problems can also be solved efficiently on graphs of bounded treewidth in many cases [68, 69].

5 Graph minors

In a long series of papers, [99, 101, 100, 105, 102, 103, 104, 107, 106, 108, 113, 114, 115, 116, 117, 109, 110, 98, 111, 112] (and others), Robertson and Seymour showed many deep results on graph minors. Some of these results will be discussed here.

A graph $G = (V, E)$ is a *minor* of a graph $H = (W, F)$, if G can be obtained from H by a series of vertex deletions, edge deletions, and edge contractions, where an edge contraction is the operation that replaces two adjacent vertices v, w by a new vertex that is adjacent to all vertices that were adjacent to v or w.

Theorem 5.1 *If $\mathcal{G}$ is a class of graphs that is closed under taking of minors, then there exists a finite set of graphs $ob(\mathcal{G})$, the* obstruction set *of $\mathcal{G}$, such for every graph H: $H \in \mathcal{G}$, if and only if no element of $ob(\mathcal{G})$ is a minor of H.*

The theorem, formerly known as Wagner's conjecture, is equivalent to stating that every class of graphs has a finite number of minor-minimal elements. An example of an obstruction set is the set $\{K_5, K_{3,3}\}$ for the minor closed class of planar graphs, or the set $\{K_3\}$ for the minor closed class of forests. There are several results giving the obstruction sets of specific minor closed classes of graphs, e.g., the obstruction set of graphs of treewidth two is $\{K_4\}$; see [11, 120] for the obstruction set of graphs of treewidth 3, and [84] for the obstruction sets of graphs of pathwidth 1, respectively 2. The size of the obstruction sets can grow very fast: for instance, the obstruction set of the graphs with pathwidth at most k contains at least $k!^2$ trees, each containing $\frac{5 \cdot 3^k - 1}{2}$ vertices [124]. Ramachandramurthi [96, 97] investigated the graphs with $k+1$, $k+2$ and $k+3$ vertices that belong to the obstruction sets for graphs of treewidth or pathwidth k. See, e.g., also [41, 56].

Some additional results make that Theorem 5.1 has surprising algorithmic implications. Robertson and Seymour [115] have shown that for every fixed graph H, there exists an algorithm that decides in $O(n^3)$ time whether H is a minor of a given graph G. Combining this algorithm with Theorem 5.1, we have the following result:

Theorem 5.2 (Robertson, Seymour) *If $\mathcal{G}$ is a class of graphs, closed under taking of minors, then there exists an algorithm that decides membership in $\mathcal{G}$ in $O(n^3)$ time.*

(The algorithm checks for each element in the obstruction set of $\mathcal{G}$ whether it is a minor of the input graph.) Note that the result is non-constructive in two ways: only existence of the algorithm is shown, and the algorithm only decides but does not construct solutions.

If the minor closed class of graphs $\mathcal{G}$ does not contain all planar graphs, then a linear time algorithm is possible.

Theorem 5.3 (Robertson, Seymour [101]) *For any planar graph H, there is a constant c_H, such that every graph with no minor isomorphic to H has treewidth at most c_H.*

There are planar graphs with arbitrary large treewidth, and planarity is preserved under minor taking, thus it is not possible to prove a variant of Theorem 5.3 for non-planar graphs H. If H is a forest, then there exists a similar upper bound c_H on the *pathwidth* of graphs that do not contain H as a minor (see [16, 42]). In [118], it is shown that one can take $c_H = 20^{2(2|V_H|+4|E_H|)^5}$. A similar type of bound was proved by Gorbunov [72]. In some special cases, one can prove better bounds. For instance, for $H = C_k$, the cycle with k vertices, then one can take $c_H = k - 1$ [64]. If $H = K_{2,k}$, then one can take $c_H = 2k - 2$ [35]. Other special cases are discussed in [20, 23, 35].

As on graphs of bounded treewidth, one can check for any fixed graph H whether it is a minor of input graph G in linear time (e.g., for fixed H, the property can be formulated in monadic second order logic, see e.g., [12]), we have the following result.

Theorem 5.4 *If $\mathcal{G}$ is a class of graphs, closed under taking of minors and that does not contain all planar graphs, then there exists an algorithm that decides membership in $\mathcal{G}$ in $O(n)$ time.*

(Let planar graph $H \notin \mathcal{G}$. Check whether the treewidth of G is at most c_H (linear time by the algorithm of [24]). If not, answer no. If so, test minorship of all graphs in the obstruction set — this can now be done in linear time.)

In some cases, self-reduction can help to overcome the non-constructive aspects of this theory. A general technique has been established by Fellows and Langston [67].

An application of the theorem recently arose in the area of distributed computing, specifically interval routing. The class of graphs for which k-interval routing schemes exist under varying edge lengths, k-IRS is closed under taking of minors, hence there exists a linear time checkable characterization for each fixed k. (See [35] for precise definitions and more results.) Still, no actual characterization is known.

Several applications for problems from graph layout, VLSI-design, and graph theory have been found by Fellows and Langston. See e.g., [62, 63, 66, 65, 15].

6 Fixed parameter complexity

Some problems are not (known to be) linear time solvable when restricted to graphs of bounded treewidth. The following behaviors can often be observed: the problem is NP-complete; the problem can be solved in $O(f(k)n^{g(k)})$ time (k the treewidth, f, g some functions growing with k); the problem can be solved in time $O(f(k)n^c)$ time (c a constant, f a function). More generally, this type of behavior can be seen in *parameterized problems*: part of the instance is distinguished as the 'parameter' often an integer, which might be small in practice. To distinguish between the second and third type of behavior, Downey and Fellows introduced the theory of *fixed parameter complexity* [58, 59, 57, 1]. Hereto, they introduced the notion of parameterized language (or problem): a subset $L \subseteq \Sigma^* \times \Sigma^*$ for some fixed alphabet Σ. The second part of the input is called the parameter; we are interested in what happens if this parameter is 'small'. Downey and Fellows also define a notion of reduction between parameterized languages, the class of *fixed parameterized tractable problems* FPT (the class of parameterized languages

L for which there exists an algorithm that decides whether $< x, k > \in L$ in $f(k)|x|^c$ time, f a function and c a constant), and a notion of reduction between parameterized languages (that preserves fixed parameterized tractability). Then they introduce a hierarchy of complexity classes $FTP \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[i] \subseteq \cdots W[P]$ of parameterized problems. Classes $W[i]$, $W[P]$ are defined in terms of reductions to certain parameterized problems on Boolean circuits. It is conjectured that the hierarchy is proper. So, hardness for $W[1]$ or any larger class means for a problem that it is unlikely that there exists an algorithm for it with time complexity of the form $O(f(k)n^c)$.

As an example: the treewidth of a graph is never larger than its bandwidth. Bandwidth is solvable in $O(f(K)n^K)$ time, K the bandwidth to obtain [75]. Bandwidth is hard for all $W[i]$, $i \in \mathbf{N}$ [28], and hence it is unlikely that the bounded treewidth of yes-instances will help to get an e.g., a linear time algorithm for bandwidth for fixed k, even with the help of tree decompositions. Other graph problems where yes-instances have bounded treewidth, and which are hard for $W[1]$ or a larger class, can be found in [28, 27, 29, 82, 83].

Postscript

I want to thank all who cooperated with me and informed me on all kinds of treewidth related topics in the past years, and to apologize to those whose work I forgot to mention here.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for $W[P]$ and PSPACE analogues. *Annals of Pure and Applied Logic*, 73:235–276, 1995.

[2] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In N. Robertson and P. Seymour, editors, *Proceedings of the AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

[3] S. Arnborg. Some PSPACE-complete logic decision problems that become linear time solvable on formula graphs of bounded treewidth. Manuscript, 1991.

[4] S. Arnborg. Graph decompositions and tree automata in reasoning with uncertainty. *J. Expt. Theor. Artif. Intell.*, 5:335–357, 1993.

[5] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[6] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, 1993.

[7] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12:308–340, 1991.

[8] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.

[9] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Disc. Appl. Math.*, 23:11–24, 1989.

[10] S. Arnborg and A. Proskurowski. Canonical representations of partial 2- and 3-trees. *BIT*, 32:197–214, 1992.

[11] S. Arnborg, A. Proskurowski, and D. G. Corneil. Forbidden minors characterization of partial 3-trees. *Disc. Math.*, 80:1–19, 1990.

[12] S. Arnborg, A. Proskurowski, and D. Seese. Monadic second order logic, tree automata and forbidden minors. In E. Börger, H. Kleine Büning, M. M. Richter, and W. Schönfeld, editors, *Proceedings 4th Workshop on Computer Science Logic, CSL'90*, pages 1–16. Springer Verlag, Lecture Notes in Computer Science, vol. 533, 1991.

[13] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8:216–235, 1987.

[14] D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). *DIMACS Ser. in Discrete Mathematics and Theoretical Computer Science*, 5:33–49, 1991.

[15] D. Bienstock and M. A. Langston. Algorithmic implications of the graph minor theorem. To appear as chapter in: Handbook of Operations Research and Management Science: Volume on Networks and Distribution, 1993.

[16] D. Bienstock, N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a forest. *J. Comb. Theory Series B*, 52:274–283, 1991.

[17] H. L. Bodlaender. NC-algorithms for graphs with small treewidth. In J. van Leeuwen, editor, *Proceedings 14th International Workshop on Graph-Theoretic Concepts in Computer Science WG'88*, pages 1–10. Springer Verlag, Lecture Notes in Computer Science, vol. 344, 1988.

[18] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11:631–643, 1990.

[19] H. L. Bodlaender. Complexity of path-forming games. *Theor. Comp. Sc.*, 110:215–245, 1993.

[20] H. L. Bodlaender. On linear time minor tests with depth first search. *J. Algorithms*, 14:1–23, 1993.

[21] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[22] H. L. Bodlaender. Dynamic algorithms for graphs with treewidth 2. In *Proceedings 19th International Workshop on Graph-Theoretic Concepts in Computer Science WG'93*, pages 112–124, Berlin, 1994. Springer Verlag.

[23] H. L. Bodlaender. On disjoint cycles. *Int. J. Found. Computer Science*, 5(1):59–68, 1994.

[24] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[25] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. Technical Report UU-CS-1996-02, Department of Computer Science, Utrecht University, Utrecht, 1996.

[26] H. L. Bodlaender and B. de Fluiter. Reduction algorithms for constructing solutions in graphs with small treewidth. In J.-Y. Cai and C. K. Wong, editors, *Proceedings 2nd Annual International Conference on Computing and Combinatorics, COCOON'96*, pages 199–208. Springer Verlag, Lecture Notes in Computer Science, vol. 1090, 1996.

[27] H. L. Bodlaender and J. Engelfriet. Domino treewidth. *J. Algorithms*, 24:94–123, 1997.

[28] H. L. Bodlaender, M. R. Fellows, and M. Hallett. Beyond *NP*-completeness for problems of bounded width: Hardness for the W hierarchy. In *Proceedings of the 26th Annual Symposium on Theory of Computing*, pages 449–458, New York, 1994. ACM Press.

[29] H. L. Bodlaender, M. R. Fellows, M. T. Hallett, H. T. Wareham, and T. J. Warnow. The hardness of problems on thin colored graphs. Technical Report UU-CS-1995-36, Department of Computer Science, Utrecht University, Utrecht, 1995.

[30] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, and minimum elimination tree height. *J. Algorithms*, 18:238–255, 1995.

[31] H. L. Bodlaender and T. Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. In Z. Fülöp and F. Gécseg, editors, *Proceedings 22nd International Colloquium on Automata, Languages and Programming*, pages 268–279, Berlin, 1995. Springer-Verlag, Lecture Notes in Computer Science 944. To appear in SIAM J. Computing, 1997.

[32] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

[33] H. L. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM J. Disc. Meth.*, 8(4):606–616, 1995.

[34] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Meth.*, 6:181–188, 1993.

[35] H. L. Bodlaender, R. B. Tan, D. M. Thilikos, and J. van Leeuwen. On interval routing schemes and treewidth. In M. Nagl, editor, *Proceedings 21th International Workshop on Graph Theoretic Concepts in Computer Science WG'95*, pages 181–186. Springer Verlag, Lecture Notes in Computer Science, vol. 1017, 1995.

[36] H. L. Bodlaender and D. M. Thilikos. Treewidth and small separators for graphs with small chordality. Technical Report UU-CS-1995-02, Department of Computer Science, Utrecht University, Utrecht, 1995. To appear in Disc. Appl. Math.

[37] H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings 24th International Colloquium on Automata, Languages, and Programming*, pages 627–637. Springer Verlag, Lecture Notes in Computer Science, vol. 1256, 1997.

[38] R. B. Borie. Generation of polynomial-time algorithms for some optimization problems on tree-decomposable graphs. *Algorithmica*, 14:123–137, 1995.

[39] R. B. Borie, R. G. Parker, and C. A. Tovey. Deterministic decomposition of recursive graph classes. *SIAM J. Disc. Meth.*, 4:481 – 501, 1991.

[40] H. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill in and tree width for distance hereditary graphs. In U. Faigle and C. Hoede, editors, *Scientific program 5th Twente Workshop on Graphs and Combinatorial Optimization*, pages 48–50, 1997.

[41] K. Cattell, M. J. Dinneen, and M. R. Fellows. Obstructions to within a few vertices or edges of acyclic. In *Proceedings 4th Int. Workshop on Algorithms*

and Data Structures. Springer Verlag, Lecture Notes in Computer Science, vol. 955, 1995.

[42] K. Cattell, M. J. Dinneen, and M. R. Fellows. A simple linear-time algorithm for finding path-decompositions of small width. *Inform. Proc. Letters*, 57:197–204, 1996.

[43] N. Chandrasekharan and S. T. Hedetniemi. Fast parallel algorithms for tree decomposing and parsing partial k-trees. In *Proc. 26th Annual Allerton Conference on Communication, Control, and Computing*, Urbana-Champaign, Illinois, 1988.

[44] S. Chaudhuri and C. D. Zaroliagis. Optimal parallel shortest paths in small treewidth digraphs. In P. Spirakis, editor, *Proceedings 3rd Annual European Symposium on Algorithms ESA'95*, pages 31–45. lncs979, 1995.

[45] R. F. Cohen, S. Sairam, R. Tamassia, and J. S. Vitter. Dynamic algorithms for optimization problems in bounded tree-width graphs. In *Proceedings of the 3rd Conference on Integer Programming and Combinatorial Optimization*, pages 99–112, 1993.

[46] W. Cook and P. D. Seymour. An algorithm for the ring-routing problem. Bellcore technical memorandum, Bellcore, 1993.

[47] B. Courcelle. The monadic second-order logic of graphs II: Infinite graphs of bounded width. *Mathematical Systems Theory*, 21:187–221, 1989.

[48] B. Courcelle. Graph rewriting: an algebraic and logical approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, volume B*, pages 192–242, Amsterdam, 1990. North Holland Publ. Comp.

[49] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[50] B. Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theor. Comp. Sc.*, 80:153–202, 1991.

[51] B. Courcelle. The monadic second-order logic of graphs III: Treewidth, forbidden minors and complexity issues. *Informatique Théorique*, 26:257–286, 1992.

[52] B. Courcelle and J. Lagergren. Equivalent definitions of recognizability for sets of graphs of bounded tree-width. *Mathematical Structures in Computer Science*, 6:141–166, 1996.

[53] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comp. Sc.*, 109:49–82, 1993.

[54] B. de Fluiter. *Algorithms for Graphs of Small Treewidth.* PhD thesis, Utrecht University, 1997.

[55] B. de Fluiter and H. L. Bodlaender. Parallel algorithms for treewidth two, 1997. To appear in Proceedings WG'97.

[56] M. J. Dinneen. *Bounded Combinatorial Width and Forbidden Substructures.* PhD thesis, University of Victoria, 1995.

[57] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory*, pages 191–226. Cambridge University Press, 1993.

[58] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24:873–921, 1995.

[59] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theor. Comp. Sc.*, 141:109–131, 1995.

[60] J. A. Ellis, I. H. Sudborough, and J. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113:50–79, 1994.

[61] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In Proceedings SODA'95, 1995.

[62] M. R. Fellows and M. A. Langston. Nonconstructive advances in polynomial-time complexity. *Inform. Proc. Letters*, 26:157–162, 1987.

[63] M. R. Fellows and M. A. Langston. Nonconstructive tools for proving polynomial-time decidability. *J. ACM*, 35:727–739, 1988.

[64] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. In *Proceedings of the 21rd Annual Symposium on Theory of Computing*, pages 501–512, 1989.

[65] M. R. Fellows and M. A. Langston. Fast search algorithms for layout permutation problems. *Int. J. on Computer Aided VLSI Design*, 3:325–340, 1991.

[66] M. R. Fellows and M. A. Langston. On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM J. Disc. Meth.*, 5:117–126, 1992.

[67] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. *J. Comp. Syst. Sc.*, 49:769–779, 1994.

[68] D. Fernández-Baca and A. Medipalli. Parametric module allocation on partial k-trees. *IEEE Trans. on Computers*, 42:738–742, 1993.

[69] D. Fernández-Baca and G. Slutzki. Parametic problems on graphs of bounded treewidth. *J. Algorithms*, 16:408–430, 1994.

[70] G. N. Frederickson. Maintaining regular properties dynamically in k-terminal graphs. Manuscript. To appear in *Algorithmica.*, 1993.

[71] R. Garbe. Tree-width and path-width of comparability graphs of interval orders. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Proceedings 20th International Workshop on Graph Theoretic Concepts in Computer Science WG'94*, pages 26–37. Springer Verlag, Lecture Notes in Computer Science, vol. 903, 1995.

[72] K. Y. Gorbunov. An estimate of the tree-width of a graph which has not a given planar grid as a minor. Manuscript, 1993.

[73] D. Granot and D. Skorin-Kapov. NC algorithms for recognizing partial 2-trees and 3-trees. *SIAM J. Disc. Meth.*, 4(3):342–354, 1991.

[74] A. Gupta and N. Nishimura. The complexity of subgraph isomorphism: Duality results for graphs of bounded path- and tree-width. Technical Report CS-95-14, University of Waterloo, Computer Science Department, Waterloo, Ontario, Canada, 1995.

[75] E. M. Gurari and I. H. Sudborough. Improved dynamic programming algorithms for bandwidth minimization and the mincut linear arrangement problem. *J. Algorithms*, 5:531–546, 1984.

[76] J. Gustedt. On the pathwidth of chordal graphs. *Disc. Appl. Math.*, 45(3):233–248, 1993.

[77] M. Habib and R. H. Möhring. Treewidth of cocomparability graphs and a new order-theoretic parameter. *ORDER*, 1:47–60, 1994.

[78] T. Hagerup. Dynamic algorithms for graphs of bounded treewidth. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings 24th International Colloquium on Automata, Languages, and Programming*, pages 292–302. Springer Verlag, Lecture Notes in Computer Science, vol. 1256, 1997.

[79] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizations of k-terminal flow networks and computing network flows in partial k-trees. In *Proceedings SODA'95*, 1995.

[80] K. Jansen and P. Scheffler. Generalized coloring for tree-like graphs. In *Proceedings 18th International Workshop on Graph-Theoretic Concepts in Computer Science WG'92*, pages 50–59, Berlin, 1993. Springer Verlag, Lecture Notes in Computer Science, vol. 657.

[81] D. Kaller. Definability equals recognizability of partial 3-trees. In *Proceedings 22nd International Workshop on Graph-Theoretic Concepts in Computer Science WG'96*, pages 239–253. Springer Verlag, Lecture Notes in Computer Science, vol. 1197, 1997.

[82] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM J. Comput.*, 25:540–561, 1996.

[83] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In *Proceedings of the 35th annual symposium on Foundations of Computer Science (FOCS)*, pages 780–791. IEEE Computer Science Press, 1994.

[84] N. G. Kinnersley and M. A. Langston. Obstruction set isolation for the gate matrix layout problem. *Disc. Appl. Math.*, 54:169–213, 1994.

[85] T. Kloks. Treewidth of circle graphs. In *Proceedings Forth International Symposium on Algorithms and Computation, ISAAC'93*, pages 108–117, Berlin, 1993. Springer Verlag, Lecture Notes in Computer Science, vol. 762.

[86] T. Kloks. *Treewidth. Computations and Approximations*. Lecture Notes in Computer Science, Vol. 842. Springer-Verlag, Berlin, 1994.

[87] T. Kloks and D. Kratsch. Treewidth of chordal bipartite graphs. *J. Algorithms*, 19:266–281, 1995.

[88] E. Korach and N. Solel. Linear time algorithm for minimum weight Steiner tree in graphs with bounded treewidth. Manuscript, 1990.

[89] A. Kornai and Z. Tuza. Narrowness, pathwidth, and their application in natural language processing. *Disc. Appl. Math.*, 36:87–92, 1992.

[90] J. Lagergren. Efficient parallel algorithms for graphs of bounded tree-width. *J. Algorithms*, 20:20–44, 1996.

[91] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society. Series B (Methodological)*, 50:157–224, 1988.

[92] S. Mahajan and J. G. Peters. Regularity and locality in k-terminal graphs. *Disc. Appl. Math.*, 54:229–250, 1994.

[93] E. Mata-Montero. Resilience of partial k-tree networks with edge and node failures. *Networks*, 21:321–344, 1991.

[94] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Disc. Math.*, 108:343–364, 1992.

[95] A. Parra. *Structural and Algorithmic Aspects of Chordal Graph Embeddings.* PhD thesis, Technical University Berlin, 1996.

[96] S. Ramachandramurthi. *Algorithms for VLSI Layout Based on Graph Width Metrics.* PhD thesis, Computer Science Department, University of Tennessee, Knoxville, Tennessee, USA, 1994.

[97] S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Meth.*, 10:146–157, 1997.

[98] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. In prepartion.

[99] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory Series B*, 35:39–61, 1983.

[100] N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory Series B*, 36:49–64, 1984.

[101] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[102] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory Series B*, 41:92–114, 1986.

[103] N. Robertson and P. D. Seymour. Graph minors. VI. Disjoint paths across a disc. *J. Comb. Theory Series B*, 41:115–138, 1986.

[104] N. Robertson and P. D. Seymour. Graph minors. VII. Disjoint paths on a surface. *J. Comb. Theory Series B*, 45:212–254, 1988.

[105] N. Robertson and P. D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Comb. Theory Series B*, 48:227–254, 1990.

[106] N. Robertson and P. D. Seymour. Graph minors. IX. Disjoint crossed paths. *J. Comb. Theory Series B*, 49:40–77, 1990.

[107] N. Robertson and P. D. Seymour. Graph minors. VIII. A Kuratowski theorem for general surfaces. *J. Comb. Theory Series B*, 48:255–288, 1990.

[108] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory Series B*, 52:153–190, 1991.

[109] N. Robertson and P. D. Seymour. Graph minors. XVI. Excluding a non-planar graph. Manuscript, 1991.

[110] N. Robertson and P. D. Seymour. Graph minors. XVII. Taming a vortex. Manuscript, 1991.

[111] N. Robertson and P. D. Seymour. Graph minors. XXI. Graphs woth unique linkages. Manuscript, 1992.

[112] N. Robertson and P. D. Seymour. Graph minors. XXII. Irrelevant vertices in linkage problems. Manuscript, 1992.

[113] N. Robertson and P. D. Seymour. Graph minors. XI. Distance on a surface. *J. Comb. Theory Series B*, 60:72–106, 1994.

[114] N. Robertson and P. D. Seymour. Graph minors. XII. Excluding a non-planar graph. *J. Comb. Theory Series B*, 64:240–272, 1995.

[115] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory Series B*, 63:65–110, 1995.

[116] N. Robertson and P. D. Seymour. Graph minors. XIV. Extending an embedding. *J. Comb. Theory Series B*, 65:23–50, 1995.

[117] N. Robertson and P. D. Seymour. Graph minors. XV. Giant steps. *J. Comb. Theory Series B*, 68:112–148, 1996.

[118] N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory Series B*, 62:323–348, 1994.

[119] D. P. Sanders. On linear recognition of tree-width at most four. *SIAM J. Disc. Meth.*, 9(1):101–117, 1996.

[120] A. Satyanarayana and L. Tung. A characterization of partial 3-trees. *Networks*, 20:299–322, 1990.

[121] P. Scheffler. A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Report 396/1994, TU Berlin, Fachbereich Mathematik, Berlin, 1994.

[122] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[123] R. Sundaram, K. Sher Singh, and C. Pandu Rangan. Treewidth of circular-arc graphs. *SIAM J. Disc. Meth.*, 7:647–655, 1994.

[124] A. Takahashi, S. Ueno, and Y. Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Disc. Math.*, 127(1/3):293 – 304, 1994.

[125] J. Telle and A. Proskurowski. Efficient sets in partial k-trees. *Disc. Appl. Math.*, 44:109–117, 1993.

[126] J. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proceedings of Workshop on Algorithms and Data Structures WADS'93*, pages 610–621. Springer Verlag, Lecture Notes in Computer Science, vol. 700, 1993.

[127] M. Thorup. Structured programs have small tree-width and good register allocation. Technical Report DIKU-TR-95/18, Department of Computer Science, University of Copenhagen, Denmark, 1995. To appear in: Proceedings WG'97.

[128] E. Wanke. Bounded tree-width and LOGCFL. *J. Algorithms*, 16:470–491, 1994.

[129] T. V. Wimer. *Linear Algorithms on k-Terminal Graphs*. PhD thesis, Dept. of Computer Science, Clemson University, 1987.

[130] X. Zhou, S. Nakano, and T. Nishizeki. Edge-coloring partial k-trees. *J. Algorithms*, 21:598–617, 1996.