Dynamic Programming on Graphs with Bounded Treewidth*

Hans L. Bodlaender†
Dept. of Computer Science, University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract

In this paper we study the complexity of graph decision problems, restricted to the class of graphs with treewidth $\leq k$, (or equivalently, the class of partial k-trees), for fixed k. We introduce two classes of graph decision problems, LCC and ECC, and subclasses C-LCC, and C-ECC. We show that each problem in LCC (or C-LCC) is solvable in polynomial ($\mathcal{O}(n^C)$) time, when restricted to graphs with fixed upper bounds on the treewidth and degree; and that each problem in ECC (or C-ECC) is solvable in polynomial ($\mathcal{O}(n^C)$) time, when restricted to graphs with a fixed upper bound on the treewidth (with given corresponding tree-decomposition). Also, problems in C-LCC and C-ECC are solvable in polynomial time for graphs with a logarithmic treewidth, and in the case of C-LCC-problems, a fixed upper bound on the degree of the graph.

Also, we show for a large number of graph decision problems, their membership in LCC, ECC, C-LCC and/or C-ECC, thus showing the existence of $\mathcal{O}(n^C)$ or polynomial algorithms for these problems, restricted to the graphs with bounded treewidth (and bounded degree). In several cases, $C = 1$, hence our method gives in these cases linear algorithms.

For several NP-complete problems, and subclasses of the graphs with bounded treewidth, polynomial algorithms have been obtained. In a certain sense, the results in this paper unify these results.

Keywords: Treewidth, partial k-trees, graph decision problems, restrictions of NP-complete problems, polynomial time algorithms, dynamic programming, local condition compositions.

1 Introduction

In general it is believed that NP-complete problems cannot be solved in polynomial time. Therefore much research has been done on the complexity of subproblems of NP-complete problems.

In this paper we consider (NP-complete) graphs problems, and we pose as restriction on the graphs in the instance of the problems, that the treewidth of the graphs is bounded by a constant k. For some problems, we pose as an extra restriction that the degree of the graphs is bounded by some constant d. We prove that for large classes of (NP-complete) problems, these become solvable in polynomial time with the extra restrictions. The algorithms are polynomial in the problem-size, but exponential in k (and d).

Arnborg and Proskurowski [5] also studied the problem of the complexity of (NP-hard) graph problems on graphs with bounded treewidth, and obtained linear time algorithms for the following problems: VERTEX COVER, INDEPENDENT SET, DOMINATING SET, GRAPH K-COLORABILITY, HAMILTONIAN CIRCUIT, NETWORK RELIABILITY. The algorithms are linear in the size of the problem instance, but are exponential in the treewidth of the involved graphs. The algorithms in this paper have some similarity to the algorithms in [5], but we think the approach in our paper is more general and easier to use. For an overview, see also [1]. Independently, results of a similar type were obtained by Scheffler and Seese [29] and Arnborg, Lagergren and Seese [4].

The class of graphs with treewidth bounded by some constant k is also important for the following reason. To many well-known classes of graphs one can associate a constant k, such that each graph in the class has treewidth k or less. For example, the treewidth of a series-parallel graph or an outerplanar graph is at most 2, the treewidth of a Halin graph is 3. For an overview of results of this type, see [8]. The

*Part of this work was carried out at the Dept. of Computer Science of the University of Utrecht, with financial support from the Foundation of Computer Science (S.I.O.N.) of the Netherlands Organization for the Advancement of Pure Research (Z.W.O.), and part of it was carried out at the Lab. of Computer Science of the Massachusetts Institute of Technology, with financial support of the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

†Electronic mail-address: ..!mcvax!ruuinf!hansb.

following classes of graphs have a constant number as bound for the treewidth of the graphs in the class: trees, forests, almost trees with parameter k, graphs with bandwidth at most k, graphs with cutwidth at most k, series-parallel graphs, outerplanar graphs, Halin graphs, k-outerplanar graphs, chordal graphs with maximum clique size k, circular arc graphs with maximum clique size k, and k-bounded treepartite graphs (k a constant).

Many polynomial time algorithms have been devised for NP-complete graph problems, restricted to graphs in one of the above mentioned classes. See e.g. [6,12,13,14,15,17,18,19,21,22,24,25,31,34,35]. In [33] a general approach is taken for a certain class of problems, on series-parallel graphs (i.e. graphs with treewidth ≤ 2). In [7] also a general approach is taken, for several of the mentioned classes of graphs. One can observe that for many of these algorithms, the underlying technique is *dynamic programming.*

In some sense, this paper explains the observed similarity of the complexity results for many of the mentioned classes of graphs, and unifies several of the mentioned papers. Of course, in many cases, a better algorithm is obtained, by looking at a single problem on a more restricted class of graphs. In this paper we take a general approach, which proves membership in P for many problems for the (rather general) class of graphs with bounded treewidth.

2 Definitions and preliminary results

2.1 Graph definitions

First we introduce some notations and definitions dealing with graphs. For every undirected graph $G = (V, E)$, degree(G) denotes the maximum degree over all vertices in V. The distance between two vertices $v, w \in V$ in the graph $G = (V, E)$ is denoted by $d_G(v, w)$.

Let $G = (V, E)$ be an undirected graph and let $c \geq 0$ be an integer. The set of vertices with distance at most c to $v \in V$ is denoted by $N_c(v, G) = \{w \in V \mid d_G(v, w) \leq c\}$. The set of vertices with distance at most c to $W \subseteq V$ is denoted by $N_c(W, G) = \bigcup_{v \in W} N_c(v, G)$. The set of edges with distance at most c to $v \in V$ is denoted by $M_c(v, G) = \{(w, x) \in E \mid d_G(v, w) \leq c \wedge d_G(v, x) \leq c\}$. The set of edges with distance at most c to $W \subseteq V$ is denoted by $M_c(W, G) = \bigcup_{v \in W} M_c(v, G)$.

When there cannot be confusion over which graph G is used, G is dropped as subscript and index. Next we introduce the definition of the treewidth of a graph, introduced by Robertson and Seymour [26].

DEFINITION 2.1 Let $G = (V, E)$ be a graph. A tree-decomposition of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a family of subsets of V, $T = (I, F)$ is a tree, with the following properties:

1. $\bigcup_{i \in I} X_i = V$
2. For every edge $e = (v, w) \in E$, there is a subset X_i, $i \in I$ with $v \in X_i$ and $w \in X_i$.
3. For all $i, j, k \in I$, if j lies on the path in T from i to k, then $X_i \cup X_k \subseteq X_j$.

The treewidth of a tree-decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$. The treewidth of G, denoted by treewidth(G), is the minimum treewidth of a tree-decomposition of G, taken over all possible tree-decompositions of G.

We denote the class of graphs with treewidth at most k, by TW(k). The class of graphs with treewidth at most k, and degree at most d is denoted by TWD(k, d).

Lemma 2.1 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$ with treewidth k. Then there exists a tree-decomposition $(\{Y_i \mid i \in J\}, T' = (J, F'))$ of G with treewidth $\leq k$ and $|J| \leq |V| - k + 1$.*

Independently, Arnborg, Corneil and Proskurowski [2] and Robertson and Seymour [26,28] have shown that there exist polynomial algorithms to test whether a graph has treewidth $\leq k$ for any given fixed k. The general problem of deciding the treewidth of a graph is NP-complete [2]. The algorithms in [2], [26] can also be used to actually yield tree-decompositions with the desired treewidth, if such exist. Robertson and Seymour [28] give a non-constructive proof, that an $\mathcal{O}(n^2)$ algorithm exists to test whether a graph has treewidth $\leq k$. It can be modified to an $\mathcal{O}(n^3)$ algorithm that finds the corresponding tree-decomposition. We use the following variant of these results.

Theorem 2.2 *For all k, there exists an algorithm, that finds for each graph $G = (V, E)$ with treewidth(G)$\leq$ k, in polynomial time a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G with treewidth at most k, and $|I| \leq |V| - k + 1$.*

2.2 Algebraic definitions

Next we recall and introduce some algebraic notions and definitions. Recall that a monoid is a 3-tuple $(M, \oplus, 0)$, where M is a non-empty set, $\oplus$ is an associative binary composition on this set, and for all $a \in M: a \oplus 0 = 0 \oplus a = a$ (see e.g. [20]). For $a_1, \ldots, a_n \in M$, with $(M, \oplus, 0)$ a commutative monoid, we denote $\bigoplus_{1 \leq i \leq n} a_i = a_1 \oplus a_2 \oplus \cdots \oplus a_n$.

DEFINITION 2.2 A totally ordered commutative monoid (tocm) is a 4-tuple $(M, \oplus, 0, \leq)$, where $(M, \oplus, 0)$ is a commutative monoid and $\leq$ is a binary relation on M, which induces a total ordering on M. We say that a tocm is a consistent totally ordered commutative monoid (ctocm), if for all $a, b, c \in M: a \leq b \Rightarrow a \oplus c \leq b \oplus c$.

For our purposes, important examples of commutative monoids are: $(N, +, 0)$, $(N, \cdot, 1)$, $(Z, +, 0)$, $(\{\underline{\text{true}},\underline{\text{false}}\}, \vee, \underline{\text{false}})$, $(\{\underline{\text{true}},\underline{\text{false}}\}, \wedge, \underline{\text{true}})$. Important examples of ctocm's are: $(N, +, 0, \leq)$, $(N, +, 0, \geq)$, $(Z, +, 0, \leq)$, $(Z, +, 0, \geq)$, $(\{\underline{\text{true}},\underline{\text{false}}\}, \vee, \underline{\text{false}}, \preceq)$, $(\{\underline{\text{true}},\underline{\text{false}}\}, \wedge, \underline{\text{true}}, \preceq)$, and $\preceq$ is one of the 2 possible total orderings on $\{\underline{\text{true}},\underline{\text{false}}\}$.

2.3 Other notations

For functions $f: X \to Y$ and subsets $Z \subseteq X$, we denote the restriction of f to Z by $f|_Z$, i.e. $f|_Z: Z \to Y$ and $\forall z \in Z: f|_Z(z) = f(z)$.

2.4 Graph decision problems

In this section we give a number of definitions, dealing with decision problems on graphs. We view a decision problem Π as a 3-tuple (D_Π, Y_Π, s_Π), with D_Π the set of instances of Π, $Y_\Pi \subseteq D_\Pi$ the set of instances of Π that yield the answer *'yes'* to problem Π, and s_Π is a function $D_\Pi \to N$, giving each instance $D \in D_\Pi$ of Π a size $s_\Pi(D)$. In general, the s_Π's will be very natural measures of the size of the instances.

DEFINITION 2.3 A decision problem $\Pi = (D_\Pi, Y_\Pi, s_\Pi)$ is a graph decision problem, if each instance $D \in D_\Pi$ can be written as a 2-tuple $D = (G_D, I_D)$, where $G_D = (V_D, E_D)$ is an undirected graph and $s_\Pi(D) \geq \max(|V_D|, |E_D|)$. ($I_D$ must contain all other information of the instance D.)

Note that I_D may be empty, for instance if Π = HAMILTONIAN CIRCUIT. Directed graphs G' can be handled by using the undirected graph G_D, obtained by ignoring the direction of the edges in G', and letting I_D contain in some coded form all information on the directions of the edges. We now give a formal definition of the natural way of restricting a graph problem to a class of graphs, and a variant of the notion of polynomial transformation of decision problems (see e.g. [16], p.34), for graph decision problems. Note that the graphs G_D in instances $D = (G_D, I_D)$, do not change under a gp-transformation.

DEFINITION 2.4 For a class of graphs Θ, and a graph decision problem $\Pi = (D_\Pi, Y_\Pi, s_\Pi)$, Π, restricted to Θ, is the graph decision problem $\Pi|_\Theta = (D_{\Pi|_\Theta}, Y_{\Pi|_\Theta}, X_{\Pi|_\Theta})$, where $D_{\Pi|_\Theta} = \{(G, I) \in D_\Pi \mid G \in \Theta\}$, $Y_{\Pi|_\Theta} = \{(G, I) \in D_\Pi \mid G \in \Theta\} = Y_\Pi \cap D_{\Pi|_\Theta}$ and $s_{\Pi|_\Theta} = s_\Pi|_{D_{\Pi|_\Theta}}$.

DEFINITION 2.5 A graph-invariant polynomial transformation (or, in short: a gp-transformation) of a graph decision problem $\Pi_1 = (D_1, Y_1, s_1)$ to a graph decision problem $\Pi_2 = (D_2, Y_2, s_2)$ is a function $f: D_1 \to D_2$, satisfying:

1. if $(G, I) \in D_1$ and $f((G, I)) = (H, J) \in D_2$, then $G = H$.
2. f can be computed in time, polynomial in $s(D)$ (by a deterministic Turing machine, or some equivalent machine model)
3. for all $D \in D_1: D \in Y_1 \Leftrightarrow f(D) \in Y_2$

4. there is a polynomial p, such that for all $D \in D_1$, $s(f(D)) \leq p(s(D))$

Theorem 2.3 *Let Π_1, Π_2 be graph decision problems and let there exist a gp-transformation of Π_1 to Π_2. Let Θ be a class of graphs.*

1. *If $\Pi_2|_\Theta \in P$, then $\Pi_1|_\Theta \in P$.*
2. *If $\Pi_1|_\Theta$ is NP-complete, then $\Pi_2|_\Theta$ is NP-complete.*

2.5 LCC and ECC

In this section we give more-or-less informal definitions for the classes of graph problems LCC and ECC. The full, lengthy definitions can be found in the full paper [9].

A basic LCC problem is a problem of the form:

INSTANCE: Graph $G = (V, E)$, finite sets X, Y, subsets $R_1 \subseteq M^1, \ldots, R_m \subseteq M^m$, element $K \in M^{m+1}$, other information I.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

- $\forall p, 1 \leq p \leq m : \bigoplus^p_{v \in V} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \in R_p$
- $\bigoplus^{m+1}_{v \in V} val_{m+1}(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \preceq K$

where $(M^1, \oplus^1, 0^1), \cdots, (M^m, \oplus^m, 0^m)$ are commutative monoids, $(M^{m+1}, \oplus^{m+1}, 0^{m+1}, \preceq)$ is a ctocm, the operations "$\oplus$", "$\in R_p$", "$\preceq$" and the function val_p can be computed in polynomial time where necessary, and for $p \neq m+1$, and $S \subseteq V$, the number of different values, that $val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ can assume over all possible f and g, is polynomially bounded (by a polynomial p_p in $s((G, (X, Y, R_1, \ldots, R_m, I)))$).

DEFINITION 2.6 Let Π be a graph decision problem. We say that Π is a local condition composition problem, if and only if there exists a basic local condition composition problem Π' and a gp-transformation from Π to Π'. The class of local condition composition problems is denoted by LCC.

A subclass of LCC is the class of the edge condition composition problems, or ECC. Instead of letting *val*-functions work on the values of f and g in "local" parts of G, like $N_c(v)$ and $M_c(v)$, now the *val*-functions work on the values of $f(v)$, $f(w)$ and $g((v, w))$ for single edges $(v, w) \in E$. This difference makes that ECC is more restricted than LCC (unless P=NP). On the other hand, no restrictions on the degree of the graphs are longer necessary for ECC-problems, in order to obtain polynomial algorithms for the problems, restricted to graphs with constant bounded treewidth. A basic ECC-problem has the following form (we have similar restrictions as in the case of LCC).

INSTANCE: Graph $G = (V, E)$, finite sets X, Y, subsets $R_1 \subseteq M^1, \ldots, R_m \subseteq M^m$, element $K \in M^{m+1}$, other information I.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

- $\forall p, 1 \leq p \leq m : \bigoplus^p_{(v,w) \in E} val_p(D, (v, w), f(v), f(w), g(v, w)) \in R_p$
- $\bigoplus^{m+1}_{(v,w) \in E} val_{m+1}(D, (v, w), f(v), f(w), g((v, w))) \preceq K$

DEFINITION 2.7 Let Π be a graph decision problem. We say that Π is an edge condition composition problem, if and only if there exists a basic edge condition composition problem Π' and a gp-transformation from Π to Π'. The class of edge condition composition problems is denoted by ECC.

Theorem 2.4 *$ECC \subseteq LCC$.*

We give some examples of the type of conditions that can be expressed as $\bigoplus_{v \in V}^{p} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \in R_p$ or $\bigoplus_{(v,w) \in E}^{p} val_p(D, (v,w), f(v), f(w), g((v,w))) \in R_p$.

A condition like: "$\forall v \in V$, some property of $(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ holds", can be expressed with use of the commutative monoid $(\{\underline{\text{true}}, \underline{\text{false}}\}, \wedge, \underline{\text{true}})$. One can express the required property as a val-function to $\{\underline{\text{true}},\underline{\text{false}}\}$. Finally, one must choose the respective set $R_p = \{\underline{\text{true}}\}$.

Similarly, a condition like: "$\exists v \in V$, some property of $(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ holds", can be expressed with use of the commutative monoid $(\{\underline{\text{true}}, \underline{\text{false}}\}, \vee, \underline{\text{false}})$.

By choosing the commutative monoid $(N, +, 0)$ or $(Z, +, 0)$, conditions like

$$\sum_{v \in V} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \in R_p$$

, (with R_p a subset of N or Z) can be expressed. Note that, except for the case that $p = m + 1$, one must have that

$$\max_{V' \subseteq V} | \sum_{v \in V'} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})|$$

must be bounded polynomially in $s(D)$.

By using one of the ctocm's $(N, +, 0, \leq)$, $(Z, +, 0, \leq)$, $(N, +, 0, \geq)$, or $(Z, +, 0, \geq)$, one can express conditions like

$$\sum_{v \in V} val_{m+1}(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \leq K \text{ or } \geq K.$$

Here only the number of *bits* needed to express the values of val_{m+1}, must be bounded by a polynomial in $s(D)$.

Conditions like $\bigoplus_{(v,w) \in E}^{p} val_p(D, (v,w), f(v), f(w), g(e))$ can also be expressed, while writing the problem as a basic LCC-problem. Similarly, one can express a condition of the type $\bigoplus_{v \in V}^{p} val_p(D, v, f(v))$, while writing the problem as a basic ECC-problem. (Use mappings $V \to E$, or $E \to V$, where vertices are mapped to neighboring edges, or vice-versa. (We have to assume that G does not have isolated vertices.))

3 Polynomial time algorithms for LCC-problems and ECC-problems on graphs with bounded treewidth

In this section we give a general method to obtain polynomial time algorithms for (basic) LCC-problems, restricted to a class of graphs with bounded treewidth, and bounded degree, and for (basic) ECC-problems, restricted to a class of graphs with bounded treewidth.

Lemma 3.1 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$. Let $k \in I$ be on the path from $i \in I$ to $j \in I$ in T. Then:*

1. $N_c(X_i) \cap N_c(X_i) \subseteq N_c(X_k)$
2. $M_c(X_i) \cap M_c(X_i) \subseteq M_c(X_k)$.

Theorem 3.2 *Let $k, d \in N$. Let Θ be a class of graphs, with $\Theta \subseteq TWD(k, d)$, (i.e. every graph G in Θ has treewidth k or less and degree d or less). Let Π be a basic LCC problem. Then $\Pi|_\Theta \in P$, i.e. there exists a polynomial algorithm for Π, when restricted to the graphs with treewidth $\leq k$ and degree $\leq d$.*

Proof. Suppose m, c, $(M^1, \oplus^1), \ldots, (M^m, \oplus^m), (M^{m+1}, \oplus^{M+1}, \preceq)$, $val_1, \ldots, val_{m+1}$ are as indicated by the definition of basic local condition composition problem, applied to $\Pi = (D_\Pi, Y_\Pi, s)$.

Let the algorithm work on an instance $D = (G = (V, E), (X, Y, R_1, R_2, \ldots, R_m, K, I)) \in D_\Pi$ with $G \in$TWD(k, d), i.e. the treewidth of G is at most k and the degree of G is at most d. Our algorithm starts with finding a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G, with treewidth at most k, and $|I| \leq |V| - k + 1$. We can do this in polynomial time, as indicated by theorem 2.2.

We now designate an (arbitrary) processor as "root", so we see T as a rooted tree. The set of the sons of a node $i \in I$, is denoted by sons(i); the father of i in the tree (if $i \neq$ root), is denoted by father(i), and the set of all descendants of i, including i self, is denoted by dec(i). For each $i \in I$, we number the sons of i.

Also we define for all $i \in I$:

$$Y_i = \begin{cases} X_i & \text{if } i = \text{root.} \\ \{v \in X_i \mid v \notin X_{\text{father}(i)}\}, & \text{if } i \neq \text{root.} \end{cases}$$

$$Z_i = \bigcup_{j \in \text{dec}(i)} Y_i.$$

$$W_i = \bigcup_{j \in \text{dec}(i)} X_i.$$

Lemma 3.2.1 *For all $v \in V$ there is a unique $i \in I$, with $v \in Y_i$.*

Lemma 3.2.2 *For all $i \in I$: $Z_i = \left(\bigcup_{j \in \text{sons}(i)} Z_j \cup Y_i \right)$, and $W_i = \left(\bigcup_{j \in \text{sons}(i)} W_j \cup X_i \right)$.*

Lemma 3.2.3 *Let $i \in I$, sons(i)= $\{j_1, \ldots, j_r\}$. Let $1 \leq \alpha \leq r$. Then:*

1. $N_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \cap N_c(W_{j_\alpha}) \subseteq N_c(X_i) \cap N_c(X_{j_\alpha})$.
2. $M_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \cap M_c(W_{j_\alpha}) \subseteq M_c(X_i) \cap M_c(X_{j_\alpha})$.

The expression $\sigma_p(v, f, g)$ will be used as a shorthand notation for $val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$, and $\sigma_p(S, f, g)$ as a shorthand notation for $\bigoplus_{v \in S}^p val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$, with S a subset of V and the domains of f and g contain respectively $N_c(v), N_c(S), M_c(v), M_c(S)$.

The algorithm is based on the fact, that for partial mappings $f : N_c(W_i) \to X$, $g : M_c(W_i) \to Y$, the only information that is of use for deciding whether this f and g can be extended to a solution is $f|_{N_c(X_i)}$, $g|_{M_c(X_i)}$, and $\sigma_p(Z_i, f, g)$, for $1 \leq p \leq m+1$. (This basically follows from lemma 3.1.) For f, $f' : N_c(X_i) \to X$, and g, $g' : M_c(X_i) \to Y$, with $f|_{N_c(X_i)} = f'|_{N_c(X_i)}$, $g|_{M_c(X_i)} = g'|_{M_c(X_i)}$, and $\sigma_p(Z_i, f, g) = \sigma_p(Z_i, f', g')$, for $1 \leq p \leq m$, and $\sigma_{m+1}(Z_i, f, g) \leq \sigma_{m+1}(Z_i, f', g')$, every extension of f', g' which gives a solution, is also an extension of f, g, which gives a solution.

So, the algorithm will compute for each node $i \in I$ a table, called TABLE(i), which contains $(m+3)$-tuples of the form $(f : N_c(X_i) \to X, g : M_c(X_i) \to Y, r_1, \ldots, r_{m+1})$, such that

$$\begin{aligned} &(\tilde{f} : N_c(X_i) \to X,\ \tilde{g} : M_c(X_i) \to Y,\ r_1, \ldots, r_{m+1}) \in \text{TABLE}(i) \Leftrightarrow \\ &\text{there exists } f : N_c(W_i) \to X,\ g : M_c(W_i) \to Y, \text{ with} \\ &f|_{N_c(X_i)} = \tilde{f},\ g|_{M_c(X_i)} = \tilde{g},\ \forall p,\ 1 \leq p \leq m+1,\ \sigma_p(Z_i, f, g) = r_p \\ &\text{and for all } f : N_c(W_i) \to X,\ g : M_c(W_i) \to Y, \text{ with} \\ &f|_{N_c(X_i)} = \tilde{f},\ g|_{M_c(X_i)} = \tilde{g},\ \forall p,\ 1 \leq p \leq m,\ \sigma_p(Z_i, f, g) = r_p, \\ &\text{one has : } r_{m+1} \prec \sigma_{m+1}(Z_i, f, g). \end{aligned} \tag{1}$$

Note that for each $\tilde{f} : N_c(X_i) \to X$, $\tilde{g} : M_c(X_i) \to Y$, $r_1 \in M^1$, ..., $r_m \in M^m$, there will be at most one $r_{m+1} \in M^{m+1}$, with $(\tilde{f}, \tilde{g}, r_1, \ldots, r_{m+1}) \in \text{TABLE}(i)$. r_{m+1} is the smallest value of $\sigma_{m+1}(Z_i, f, g)$, where f and g are extensions of $\tilde{f}$ and $\tilde{g}$, such that $\sigma_p(Z_i, f, g) = r_p$ for all p, $1 \leq p \leq m$.

The algorithm starts by first recursively calculating the TABLE's for all sons of i, and then calculating temporary tables TEMP(i, α), with $0 \leq \alpha \leq r$, where r is the number of sons of i. For the these temporary tables the following condition will hold, after calculation of TEMP(i, α):

$$\begin{aligned} &(\tilde{f} : N_c(X_i) \to X,\ \tilde{g} : M_c(X_i) \to Y,\ r_1, \ldots, r_{m+1}) \in \text{TEMP}(i, \alpha) \Leftrightarrow \\ &\text{there exists } f : N_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \to X,\ g : M_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \to Y, \text{ with} \end{aligned} \tag{2}$$

$$f|_{N_c(X_i)} = \tilde{f},\ g|_{M_c(X_i)} = \tilde{g},\ \forall p,\ 1 \le p \le m+1 :\ \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p$$

and for all $f : N_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \to X,\ g : M_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \to Y$, with

$$f|_{N_c(X_i)} = \tilde{f},\ g|_{M_c(X_i)} = \tilde{g},\ \forall p,\ 1 \le p \le m :\ \sigma_p(Z_{j-1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p$$

one has : $r_{m+1} \prec \sigma_{m+1}(Z_{j-1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g)$.

We claim that the following pair of subroutines compute the TABLEs and TEMPs correctly. Basically, the algorithm works as follows: first we compute (recursively) TABLE(j_α), for all sons j_α of i. Then we compute TEMP($i, 0$), by calculating $\sigma_p(Y_i, f, g)$ for every $f : N_c(X_i) \to X$, and $g : M_c(X_i) \to Y$. Then we successively compute the tables TEMP(i, α), by composing the tables TEMP($i, \alpha - 1$) and TABLE(j_α). From lemma 3.2.2 equation (1) follows from equation (2), and hence TABLE(i) can be chosen to be TEMP(i, r).

```
CALCULATE_TABLE(i):
begin
    Let j_1, ..., j_r be the sons of i.
    for α = 1 to r do CALCULATE_TABLE(j_α) enddo;
    for α = 0 to r do CALCULATE_TEMP(i,α) enddo;
    TABLE(i) := TEMP(i, r)
end.

CALCULATE_TEMP(i, α):
begin
    if α = 0
    then for all functions f : N_c(X_i) → X
         do for all functions g : M_c(X_i) → Y
            do for p = 1 to m + 1 do let r_p = σ_p(Y_i, f, g) enddo;
               put (f, g, r_1, ..., r_{m+1}) in TEMP(i, α)
            enddo
         enddo
    else for every (f, g, r_1, ..., r_{m+1}) ∈ TEMP(i, α)
         do for every (f', g', r'_1, ..., r'_{m+1}) ∈ TEMP(i, α)
            do if ∀v ∈ N_c(X_i) ∩ N_c(X_{j_α}) : f(v) = f'(v) and ∀e ∈ M_c(X_i) ∩ M_c(X_{j_α}) : g(e) = g'(e)
               then for p = 1 to m + 1 do let s_p = r_p ⊕^p r'_p enddo;
                    if there is no t ∈ M^{m+1}, with (f, g, s_1, ..., s_m, t) ∈ TEMP(i, α)
                    then put (f, g, s_1, ..., s_m, s_{m+1}) in TEMP(i, α)
                    else suppose (f, g, s_1, ..., s_m, t) ∈ TEMP(i, α);
                         if s_{m+1} ⪯ t
                         then remove (f, g, s_1, ..., s_m, t) from TEMP(i, α);
                              put ((f, g, s_1, ..., s_m, s_{m+1}) in TEMP(i, α)
                         endif
                    endif
               endif
            enddo
         enddo
    endif
end
```

Claim 3.3 *After execution of CALCULATE_TABLE(i) equation (1) holds and after execution of CALCULATE_TEMP(i, α) equation (2) holds.*

The following claim shows how to find the answer to the question, whether $D \in Y_\Pi$, or not, i.e. the answer to the problem that we are trying to solve, by 'looking it up' in TABLE(root).

Claim 3.4 *$D \in Y_\Pi$, if and only if there are $r_1 \in R_1$, $r_2 \in R_2$, ..., $r_m \in R_m$, $r_{m+1} \in M^{m+1}$, with $r_{m+1} \prec K$, and $f : N_c(X_{\text{root}}) \to X$, $g : M_c(X_{\text{root}}) \to X$, such that $(f, g, r_1, \ldots, r_m, r_{m+1}) \in$ TABLE(root).*

It follows that after calculating TABLE(root), we can determine whether $D \in Y_\Pi$ or not, by successively inspecting all entries in TABLE(root).

It remains us to show that the algorithm uses polynomial time. First note that the finding of the tree-decomposition can be done in polynomial time, by theorem 2.2. Next we claim that for each i and α, the size of the table TEMP(i, α) (and consequently, of TABLE(i)), is polynomially bounded in $s(D)$. First note that $|N_c(X_i)|$ and $|M_c(X_i)|$ are bounded by constants, say c_1 and c_2, (that are only depending on d, k and c). Further, for all $(f, g, r_1, \ldots, r_{m+1}) \in$ TEMP(i, α), one has $r_p \in \{\sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, \tilde{f}, \tilde{g}) \mid \tilde{f} : N_c(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i) \to X,\ \tilde{g} : M_c(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i) \to Y\}$. By definition (of the class of basic LCC-problems), the number of different values that r_p can assume here, is bounded by $p_p(s(D))$. Hence, the size of TEMP(i, α) is bounded by $|X|^{c_1} \cdot |Y|^{c_2} \cdot \prod_{p=1}^{m} p_p(s(D))$, which is polynomial in $s(D)$.

So each execution of the procedure CALCULATE_TEMP uses polynomial time. As this procedure is called $2|I| - 1 = \mathcal{O}(|V|)$ times, and the total remaining work in the procedure-calls of CALCULATE_TABLE is linear in $|I|$, it follows that one can compute TABLE(root) in polynomial time. Finally we note that the last step of the algorithm also can be done in polynomial time: one can test in polynomial time for each $(f, g, r_1, \ldots, r_m, r_{m+1}) \in$ TABLE(root), whether $r_1 \in R_1, \ldots, r_m \in R_m$ and $r_{m+1} \prec K$ and TABLE(root) is of polynomial size. **Q.E.D.**

For the class of the edge condition composition problems, we can prove a similar result. We can use an algorithm, very similar to the algorithm, described in the preceding proof. Here we do not need the requirement that the degree of G is bounded by some constant d. This requirement was necessary to ensure that the size of the sets $N_c(X_i)$ and $M_c(X_i)$ was bounded by some constant (only depending on k, and d, and not on $|V|$). Now however we use instead sets $N_0(X_i) = X_i$ and $M_0(X_i)$. Note that the size of these sets is bounded by $k+1$ and $\frac{1}{2}k(k+1)$ respectively, i.e. by constants, even if there is no bound on the degree of the graphs. In this way we obtain the following result:

Theorem 3.5 *Let $k \in N$. Let Θ be a class of graphs, with $\Theta \subseteq TW(k)$, (i.e. every graph G in Θ has treewidth k or less). Let Π be a basic ECC problem. Then $\Pi|_\Theta \in P$, i.e. there exists a polynomial algorithm for Π, when restricted to the graphs with treewidth $\leq k$.*

Now we state the main result of this paper.

Theorem 3.6 *(i) Let $\Pi \in$ LCC, and let $k, d \in N^+$. Let Θ be a class of graphs with $G \in \Theta \Rightarrow$ degree$(G) \leq d \wedge$ treewidth$(G) \leq k$. Then $\Pi|_\Theta \in P$.*

(ii) Let $\Pi \in$ ECC, and let $k \in N^+$. Let Θ be a class of graphs with $G \in \Theta \Rightarrow$ treewidth$(G) \leq k$. Then $\Pi|_\Theta \in P$.

Proof. The result follows directly from theorem 2.3, theorem 3.2 and theorem 3.5. **Q.E.D.**

4 Small-degree polynomial time algorithms for subclasses of LCC and ECC

There are interesting subclasses of LCC and ECC, that yield linear, quadratic, cubic or some other small-degree polynomial time algorithms (when we do not count the time, needed for finding the tree-decompositions with the required treewidth).

A basic C-LCC problem is a basic LCC problem, with the following extra conditions:

1. There are constants c_1, c_2, such that for all $D = (G, (X, Y, R_1, \ldots, R_m, K, I) \in D_\Pi$, one has $|X| \leq c_1$; $|Y| \leq c_2$.
2. The operations "$\oplus$", "$\in R_p$", "$\preceq$" and the functions val_p can be computed in constant time where necessary.
3. For $p \neq m+1$, and $S \subseteq V$, let the number of different values, that $val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ can assume over all possible f and g, be bounded by $c_p \cdot s(D)^{d_p}$. Then $\sum_{i=1}^{m} d_p \leq C - 1$.

We let the class of C-LCC problems of the graph-decision problems that have a gp-transformation f to a basic C-LCC problem, such that f can be computed in $\mathcal{O}(s(D)^C)$ time and $s(f(D)) = \mathcal{O}(s(D))$. Similarly we define the class of C-ECC problems.

It is not difficult to verify, that the algorithms of section 3 can be used for C-LCC and C-ECC problems, and use only $\mathcal{O}(s(D)^C)$ time. For instance, note that the size of TABLEs and TEMPs are now bounded by $\mathcal{O}(s(D)^{C-1})$. (The constant factor can depend on k, and for the case of C-LCC-problems also on d, but not on $|V|$ or $s(D)$). Hence, we can proof the following result, similar as in section 3.

Theorem 4.1 *(i) Let $\Pi \in$ C-LCC, and let $k, d \in N^+$. Let Θ be a class of graphs, with $G \in \Theta \Rightarrow$ degree(G)$\leq d \wedge$ treewidth(G) $\leq k$. Then there exists a linear algorithm that solves Π, restricted to Θ, assuming that each graph $G \in \Theta$ in the instances is given together with a tree-decomposition of G with treewidth $\leq k$.*

(ii) Let $\Pi \in$ C-ECC, and let $k \in N^+$. Let Θ be a class of graphs, with $G \in \Theta \Rightarrow$ treewidth(G) $\leq k$. Then there exists a linear algorithm that solves Π, restricted to Θ, assuming that each graph $G \in \Theta$ in the instances is given together with a tree-decomposition of G with treewidth $\leq k$.

For the case that the treewidth of $G = (V, E)$ is bounded by a logarithmic factor in $|V|$, one obtains polynomial algorithms for C-LCC and C-ECC problems.

Theorem 4.2 *For every constant $c > 0$, there exists a polynomial algorithm, that, given a graph $G = (V, E)$, $|V| = n$, either outputs that treewidth(G) $> c \log n$, or gives a tree-decomposition of G with treewidth $\leq 4.5c \log n$.*

Proof. Apply algorithm 3.2. of [28] to graphs with treewidth (and hence branchwidth) $\mathcal{O}(\log n)$, and then transform the branch-decomposition to a tree-decomposition, as in [27]. **Q.E.D.**

Theorem 4.3 *(i) Let $\Pi \in$ C-LCC for some $C \geq 1$, and let $d \in N^+$. Let Θ be a class of graphs, with $\exists c > 0$: $\forall G = (V, E) \in \Theta \Rightarrow$ degree(G)$\leq d \wedge$ treewidth(G) $\leq c \cdot \log(|V|)$. Then there exists a polynomial algorithm that solves Π, restricted to Θ.*

(ii) Let $\Pi \in$ C-ECC. Let Θ be a class of graphs, with $\exists c > 0$: $\forall G = (V, E) \in \Theta \Rightarrow$ treewidth(G) $\leq c \cdot \log(|V|)$. Then there exists a polynomial algorithm that solves Π, restricted to Θ.

Proof. First use the algorithm, indicated by theorem 4.2. Then use the same algorithms as in the case of the constant bounded treewidth. It follows that the size of the tables TEMP and TABLE is polynomial in the instant-size, and hence that the algorithms use polynomial time. **Q.E.D.**

5 Problems in LCC and ECC

In this section we give two examples of problems in LCC and ECC, illustrating how one can show membership in these classes. In section 6.1 a list of problems in LCC and ECC is given. For a full description of the problems, see e.g. [16].

5.1 Minimum maximal matching

Theorem 5.1 MINIMUM MAXIMAL MATCHING $\in$ 1-*LCC.*

Proof. The problem has a graph-invariant linear transformation to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \{0\}$, $Y = \{0, 1\}$, positive integer $K \leq |V|$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: there is at most one adjacent edge e, with $g(e) = 1$.
2. $\forall e = (v, w) \in V$: $g(e) = 1$ or v or w is adjacent to an edge e', with $g(e') = 1$.
3. $\sum_{e \in E} g(e) \leq K$.

(The edges with $g(e) = 1$ represent the edges in E'.) **Q.E.D.**

Theorem 5.2 MINIMUM MAXIMAL MATCHING $\in$ *ECC*.

Proof. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = E \cup \{0\}$, $Y = \{0, 1\}$, positive integer $K \leq |V|$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V : f(v)$ is an edge, adjacent to v or $f(v) = 0$.
2. $\forall e = (v, w) \in E :$ if $g(e) = 1$, then $f(v) = e$ and $f(w) = e$.
3. $\forall e = (v, w) \in E :$ if $f(v) = e$ or $f(w) = e$, then $g(e) = 1$.
4. $\forall e = (v, w) \in E : f(v) \neq 0$ or $f(w) \neq 0$ or $g(e) = 1$.
5. $\sum_{e \in E} g(e) \leq K$.

Q.E.D.

5.2 Partition into forests

Theorem 5.3 PARTITION INTO FORESTS $\in$ *ECC*.

Proof. There is a gp-transformation of PARTITION INTO FORESTS to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \{1, \ldots, K\} * \{0, \ldots, |V| - 1\} * V$, $Y = \{0\}$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall (v, w) \in E :$ if $f_1(v) = f_1(w)$, then $(f_3(v) = w$ or $f_3(w) = v)$.
2. $\forall (v, w) \in E :$ if $f_1(v) = f_1(w)$ and $f_3(v) = w$, then $f_2(v) = f_2(w) + 1$.
3. $\forall (v, w) \in E :$ if $f_1(v) = f_1(w)$ and $f_3(w) = v$, then $f_2(w) = f_2(v) + 1$.

($f_1(v)$ denotes the number of the component in which v is placed; $f_3(v)$ denotes in a certain sense the father of v in its subtree. The values of $f_2(v)$ decrease, by going up in the subtrees, and hence assure that there are no induced cycles.) Now the problem is easily seen to be in ECC. **Q.E.D.**

6 Overview of results and final remarks

6.1 Overview of results

In table 1 we give an overview of a number of results for the considered problems, showing a large number of applications of the results of this paper. For many other (NP-complete) graph decision problems, similar results can be obtained. In the first column, the complexity of the problem, restricted to TWD(k, d) (k, d fixed) is given; in the second column, the complexity when restricted to TW(k). Keys: 1, 2, 3: problem in 1-LCC, 2-LCC, 3-LCC (first column), or 1-ECC, 2-ECC or 3-ECC (2nd column), hence solvable in linear, quadratic or cubic time, for the specific classes of graphs and also solvable in polynomial time, for graphs with with logarithmic treewidth (and, in the case of column 1, degree bounded by a constant). (1): problem solvable in linear time, with other methods. See e.g. [3,29], or the full paper. P: problem in LCC or ECC, hence solvable in polynomial time. (P): Problem solvable in polynomial time, with other methods. See e.g. [3,11], or the full paper. N: Problem NP-complete. ?: Open, whether problem solvable in polynomial time or not. (Several of these problems may be easy to resolve.) (*): For Subgraph Isomorphism, the larger graph G must have the bounded treewidth. Other restrictions are given in the table.

VERTEX COVER	1	1
DOMINATING SET	1	P; (1)
DOMATIC NUMBER	1	(P)
CHROMATIC NUMBER	1	1
MONOCHROMATIC TRIANGLE	1	(1)
FEEDBACK VERTEX SET	P	P
FEEDBACK ARC SET	P	P; (1)
PARTIAL FEEDBACK ARC SET	1 (fixed L)	(1)
MINIMUM MAXIMAL MATCHING	1	P; (1)
PARTITION INTO TRIANGLES	1	P; (1)
PARTITION INTO ISOMORPHIC CONNECTED SUBGRAPHS	P	?
PARTITION INTO HAMILTONIAN SUBGRAPHS	1	P
PARTITION INTO FORESTS	P	P
PARTITION INTO CLIQUES	1	(1)
PARTITION INTO PERFECT MATCHINGS	1	P
COVERING BY CLIQUES	P	?
COVERING BY COMPLETE BIPARTITE SUBGRAPHS	P	?
CLIQUE	(1) 2	2; (1)
INDEPENDENT SET	1	1
INDUCED PATH	1 (fixed K);P; (1)	P; (1)
BALANCED COMPLETE BIPARTITE SUBGRAPH	3; (1)	3; (1)
BIPARTITE SUBGRAPH	1	1
DEGREE-BOUNDED CONNECTED SUBGRAPH	P	?
TRANSITIVE SUBGRAPH	1	(1)
CUBIC SUBGRAPH	1	P; (1)
HAMILTONIAN COMPLETION	P	P
HAMILTONIAN CIRCUIT	P; (1)	P; (1)
SUBGRAPH ISOMORPHISM for connected graphs	P (*)	N [32]
GRAPH CONTRACTABILITY to a fixed graph H	P; (1)	P; (1)
GRAPH HOMOMORPHISM to a fixed graph H	P; (1)	P; (1)
GRAPH GRUNDY NUMBERING	1	?
KERNEL	1	P; (1)
K-CLOSURE	1	1
INTERSECTION GRAPH BASIS	P	?
MAXIMUM LEAF SPANNING TREE	(1)	P; (1)
SHORTEST TOTAL PATH LENGTH SPANNING TREE	P	?
BOUNDED DIAMETER SPANNING TREE	P	P
ISOMORPHIC SPANNING TREE	P	N
BOUNDED COMPONENT SPANNING FOREST (weak version)	P	? (P, fixed B)
STEINER TREE IN GRAPHS	P; (1)	P; (1)
GRAPH PARTITIONING (weak version)	P	?
ACYCLIC PARTITIONING (weak version)	P	?
MAX CUT	1	1
MINIMUM CUT INTO BOUNDED SETS	2; (1)	2; (1)
LONGEST CIRCUIT	P; (1)	P; (1)
LONGEST PATH	P; (1)	P; (1)
CHROMATIC INDEX	1	(P)

Table 1:
Overview of complexity of problems, restricted to TWD(k, d) and TW(k). See text.

6.2 Problems, that are not in LCC, (unless P = NP)

In this section we give a number of problems, that are not in LCC (or any of its subclasses), unless P = NP. Results of this type follow directly if the problem is NP-complete, when restricted to a class of graphs with bounded treewidth (and bounded degree).

Theorem 6.1 *If $P \neq NP$, then*

1. BANDWIDTH $\notin$ *LCC.*
2. DIRECTED BANDWIDTH $\notin$ *LCC.*
3. MINIMUM CUT LINEAR ARRANGEMENT $\notin$ *LCC.*
4. WEIGHTED DIAMETER $\notin$ *ECC.*
5. BICONNECTIVITY AUGMENTATION $\notin$ *LCC.*
6. STRONG CONNECTIVITY AUGMENTATION $\notin$ *LCC.*
7. ISOMORPHIC SPANNING TREE $\notin$ *ECC.*

Proof. BANDWIDTH and DIRECTED BANDWIDTH are NP-complete for trees with degree 3. BICONNECTIVITY AUGMENTATION and STRONG CONNECTIVITY AUGMENTATION are NP-complete for graphs, without edges. WEIGHTED DIAMETER is NP-complete for trees. MINIMUM CUT LINEAR ARRANGEMENT is NP-complete for series-parallel graphs (= graphs with treewidth ≤ 2) [23]. One can show that ISOMORPHIC SPANNING TREE is NP-complete, when restricted to graphs with treewidth ≤ 3, by transformation from 3-PARTITION. From theorem 3.6 now the result follows. **Q.E.D.**

For SUBGRAPH ISOMORPHISM for connected graphs, a similar result holds. This problem is not in ECC (unless P = NP), because it is NP-complete in the case that G and H both are outerplanar graphs [32]. (Recall that each outerplanar graph has treewidth at most 2.) Note that this problem is in LCC, hence it separates the classes TW(k) and TWD(d, k) in complexity. For the OPTIMAL LINEAR ARRANGEMENT problem, Sudborough [30] announces work with Sun, which suggests that this problem is NP-complete, even when restricted to series-parallel graphs (hence OPTIMAL LINEAR ARRANGEMENT $\notin$ ECC, unless P=NP).

Also, all problems, that are not in NP, will be not in LCC or any of its subclasses.

Theorem 6.2 *$LCC \subseteq NP$.*

6.3 Final remarks

Although the formalisms may look complicated, we feel that the methods exposed in this paper will not be very difficult to use in practice; in particular, for problems in C-LCC and C-ECC and some others, it will be possible to obtain algorithms for these problems on graphs with treewidth $\leq k$, that are reasonably easy to implement, and are reasonably efficient for small values of k. Often, easy improvements on the time needed by applying the general method on specific problems can be made by using the specific characteristics of the problem. It was not the purpose of this paper to obtain the "best" algorithm for each specific problem.

In [10] it is shown, that the algorithms in this paper can be modified to NC-algorithms.

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.

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

[3] S. Arnborg, J. Lagergren, and D. Seese. Which problems are easy for tree-decomposable graphs. 1987. Ext. abstract to appear in proc. ICALP 88.

[4] S. Arnborg, J. Lagergren, and D. Seese. Which problems are easy for tree-decomposable graphs. 1988. In these proceedings.

[5] S. Arnborg and A. Proskurowski. *Linear time algorithms for NP-hard problems on graphs embedded in k-trees.* TRITA-NA-8404, Dept. of Num. Anal. and Comp. Sci., Royal Institute of Technology, Stockholm, Sweden, 1984.

[6] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *Proceedings 24th Ann. Symp. on Foundations of Computer Science*, pages 265–273, IEEE Computer Society, Los Angeles, 1983. Preliminary version.

[7] M. W. Bern, E. L. Lawler, and A. L. Wong. Why certain subgraph computations require only linear time. In *Proc. 26th Symp. on Foundations of Computer Science*, pages 117–125, 1985.

[8] H. L. Bodlaender. *Classes of Graphs with Bounded Treewidth.* Technical Report RUU-CS-86-22, Dept. Of Comp. Science, University of Utrecht, Utrecht, 1986.

[9] H. L. Bodlaender. *Dynamic programming algorithms on graphs with bounded tree-width.* Tech. Rep., Lab. for Comp. Science, M.I.T., 1987.

[10] H. L. Bodlaender. *NC-algorithms for graphs with small treewidth.* Technical Report RUU-CS-88-4, Dept. of Comp. Science, Univ. of Utrecht, Utrecht, 1988.

[11] H. L. Bodlaender. *Polynomial algorithms for Chromatic Index and Graph Isomorphism on partial k-trees.* Tech. Rep. RUU-CS-87-17, Dept. of Comp. Sc., Univ. of Utrecht, 1987.

[12] E. J. Cockayne, S. E. Goodman, and S. T. Hedetniemi. A linear algorithm for the domination number of a tree. *Inform. Proc. Letters*, 4:41–44, 1975.

[13] C. J. Colbourn and L. K. Stewart. Dominating cycles in series-parallel graphs. *Ars Combinatorica*, 19A:107–112, 1985.

[14] D. Coppersmith and U. Vishkin. Solving NP-hard problems in 'almost trees': vertex cover. *Disc. Applied Match*, 10:27–45, 1985.

[15] G. Cornuéjols, D. Naddef, and W. R. Pulleyblank. Halin graphs and the traveling salesman problem. *Math. Programming*, 26:287–294, 1983.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[17] Y. Gurevich, L. Stockmeyer, and U. Vishkin. Solving NP-hard problems on graphs that are almost trees and an application to facility location problems. *J. Assoc. Comp. Mach.*, 31:459–473, 1984.

[18] R. Hassin and A. Tamir. Efficient algorithms for optimization and selection on series-parallel graphs. *SIAM J. Alg. Disc. Meth.*, 7:379–389, 1986.

[19] S. T. Hedetniemi, R. Laskar, and J. Pfaff. *A linear algorithm for the domination number of a cactus.* Report 433, Dept. of Math. Sc., Clemson Univ., Clemson, S.C., 1983.

[20] T. W. Hungerford. *Algebra. Graduate Texts in Mathematics 73*, Springer-Verlag, New York, 1974.

[21] T. Kikuno, N. Yoshida, and Y. Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discrete Appl. Math.*, 5:299–311, 1983.

[22] R. Laskar, J. Pfaff, S. M. Hedetniemi, and S. T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM J. Alg. Disc. Meth.*, 5:420–425, 1984.

[23] B. Monien and I. Sudborough. Min cut is NP-complete for edge weighted trees. In *Proc. of Int. Conf. Automata, Languages, and Programming ICALP '86*, pages 265–274, Springer Verlag Lecture Notes in Comp. Science, Vol 226, 1986.

[24] B. Monien and I. H. Sudborough. Bandwidth-constrained NP-complete problems. In *Proc. 13th Ann. ACM Symp. on Theory of Computing*, pages 207–217, Assoc. For Computing Machinery, New York, 1981.

[25] A. Proskurowski and M. M. Sysło. *Efficient vertex- and edge-coloring of outerplanar graphs.* Report UO-CIS-TR-82-5, Dept. of Computer and Information Sc., Univ. of Oregon, Eugene, Ore., 1982.

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

[27] N. Robertson and P. Seymour. Graph minors. X. Obstructions to tree-decompositions. 1986. Manuscript.

[28] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. 1986. Manuscript.

[29] P. Scheffler and D. Seese. A combinatorial and logical approach to linear-time computability. 1986. Extended abstract.

[30] I. H. Sudborough. "Cutwidth" and related graph problems. *Bulletin of the EATCS*, 79–110, Feb. 1987.

[31] M. M. Sysło. NP-complete problems on some tree-structured graphs: a review. In M. Nagl and J. Perl, editors, *Proc. WG'83 International Workshop on Graph Theoretic Concepts in Computer Science*, pages 342–353, Univ. Verlag Rudolf Trauner, Linz, West Germany, 1983.

[32] M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theor. Comput. Science*, 17:91–97, 1982.

[33] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *J. ACM*, 29:623–641, 1982.

[34] J. Wald and C. Colbourn. Steiner trees, partial 2-trees, and minimum IFI networks. *Networks*, 13:159–167, 1983.

[35] J. Wald and C. J. Colbourn. Steiner trees in outerplanar graphs. In *Proc. 13th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, Utilitas Mathematica Publishing, Winnipeg, Ont., 1982.