LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-394

DYNAMIC PROGRAMMING ON GRAPHS WITH BOUNDED TREEWIDTH

Hans L. Bodlaender

June 1987

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

This blank page was inserted to preserve pagination.

Dynamic Programming on Graphs with Bounded Treewidth

Hans L. Bodlaender

May, 1987

Dynamic Programming on Graphs with Bounded Treewidth

Hans L. Bodlaender
Laboratory for Computer Science
Massachusetts Institute of Technology

May, 1987

[1] This work was carried out partially 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 partially 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.).

Dynamic Programming on Graphs with Bounded Treewidth

Hans L. Bodlaender, M.I.T.

May, 1987

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 upperbounds 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 upperbound 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 given corresponding tree-decomposition, and in the case of C-LCC-problems, a fixed upperbound 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 tree-width of the graphs is bounded by a constant k, (or equivalently, that the graphs are partial k-trees.) 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 will be exponential in k (and d).

Arnborg and Proskurowski [3] 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 tree-width of the involved graphs. The algorithms in this paper have some similarity to the algorithms in [3], but we think the approach in our paper is more general and easier to use. For an overview, see also [1].

Independently, Scheffler and Seese [24] introduced the notion of P-existential locally verifiable (P-ELV) properties of graphs. This notion is very similar to our notion of local condition composition problems. In [24] it is shown that for every P-ELV property P, and constants k, and d, the problem to decide whether for a given graph G with degree at most d, and treewidth at most k, P(G) holds, is solvable in polynomial time; (and, in many cases, in linear time.) We generalize from their results in two ways: our class LCC contains problems that are not expressible with P-ELV properties; and for the class of ECC-problems we do not need an upperbound on the degree of the graphs.

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 at most 5. For an overview of results of this type, see [6]. The 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 (k a constant), graphs with bandwidth at most k (k a constant), graphs with cutwidth at most k (k a constant), series-parallel graphs, outerplanar graphs, Halin graphs, k-outerplanar graphs (k a constant), chordal graphs with maximum clique size k (k a constant), circular arc graphs with maximum clique size k (k a constant), 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. [4,8,9,10,11,13,14,15,17,18,19,21,26,29,30]. In [28] a general approach is taken for a certain class of problems, on series-parallel graphs (i.e. graphs with treewidth ≤ 2). In [5] 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, and will prove 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.

DEFINITION 2.1 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 $d_G(v, w)$. When there cannot be confusion over which graph G is used, the subscript G is dropped.

DEFINITION 2.2 Let $G = (V, E)$ be an undirected graph and let $c \geq 0$ be an integer.

1. Let $v \in V$. The set of vertices with distance at most c to v is denoted by $N_c(v, G) = \{w \in V \mid d_G(v, w) \leq c\}$.

2. Let $W \subseteq V$. The set of vertices with distance at most c to W is denoted by $N_c(W,G) = \{w \in V \mid \exists v \in W : d_G(v,w) \leq c\} = \bigcup_{v \in W} N_c(v,G)$.
3. Let $v \in V$. The set of edges with distance at most $c \geq 1$ to v is denoted by $M_c(v,G) = \{(w,x) \in E \mid d_G(v,w) \leq c-1 \vee d_G(v,x) \leq c-1\} = \{(w,x) \in E \mid d_G(v,w) \leq c \wedge d_G(v,x) \leq c\}$.
4. Let $W \subseteq V$. The set of edges with distance at most $c \geq 1$ to v is denoted by $M_c(W,G) = \{(v,w) \in E \mid \exists v \in W : d_G(v,w) \leq c-1 \vee d_G(v,x) \leq c-1\} = \bigcup_{v \in W} M_c(v,G)$.
5. Let $W \subseteq V$. We denote $M_0(W,G) = \{(v,w) \in E \mid v \in W \wedge w \in W\}$.

When there cannot be confusion over which graph G is used, we drop the index G and denote: $N_c(v), N_c(W), M_c(v), M_c(W)$, etc.

Next we introduce the definition of the treewidth of a graph, introduced by Robertson and Seymour [22].

DEFINITION 2.3 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).

The class of the graphs with treewidth at most k equals the class of the partial k-trees (see e.g. [2]). To define the class of partial k-trees, we first give a recursive definition of the class of k-trees.

- K_k, the complete graph on k vertices, is a k-tree.

- If $G = (V, E)$ is a k-tree, and v_1, v_2 $\ldots v_k$ form a complete subgraph of G, then the graph $G' = (V \cup \{w\}, E \cup \{(v_i, w) \mid 1 \leq i \leq k\})$, with $w \notin V$, is also a k-tree.

A graph is a partial k-tree, if it is the subgraph of a k-tree. It is easy to show with induction, that each k-tree G has a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$, such that for each complete subgraph of G with $k+1$ vertices, there is exactly one $i \in I$, such that X_i contains all $k+1$ vertices in this complete subgraph. Now it follows that each partial k-tree has tree-width $\leq k$. The reverse relation (each graph with tree-width k is a partial k-tree), is left as an easy exercise to the reader. As a corollary the following lemma follows.

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 [22] have shown that there exist polynomial algorithms to test whether a graph has treewidth $\leq k$ for any given fixed k (or, equivalently, whether the graph is a partial k-tree). The general problem of deciding the treewidth of a graph is NP-complete [2]. The algorithms in [2], [22] can also be used to actually yield tree-decompositions with the desired treewidth, if such exist. We will use the following variant of these results (use also lemma 2.1).

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. First we recall the definition of monoids (see e.g. [16]).

DEFINITION 2.4 A monoid is a 3-tupple $(M, \oplus, 0)$, where M is a non-empty set, $\oplus$ is an associative binary composition on this set, i.e. for all $a, b, c \in M : (a \oplus b) \oplus c = a \oplus (b \oplus c)$, and 0 is an element of M such that for all $a \in M : a \oplus 0 = 0 \oplus a = a$. A monoid $(M, \oplus, 0)$ is commutative, if for all $a, b \in M : a \oplus b = b \oplus a$.

For $a_1, \ldots, a_n \in M$, with $(M, \oplus, 0)$ a commutative monoid, we denote

$$\bigoplus_{1 \leq i \leq n} a_i = a_1 \oplus a_2 \oplus \cdots \oplus a_n.$$

We also need the following algebraic structure.

DEFINITION 2.5 A totally ordered commuatative monoid (tocm) is a 4-tupple $(M, \oplus, 0, \leq)$, where $(M, \oplus, 0)$ is a commutative monoid and $\leq$ is a binary relation on M, which induces a total ordering on M:

1. for all $a, b \in M : a \leq b \vee b \leq a$
2. for all $a, b \in M : (a \leq b \wedge b \leq a) \Rightarrow a = b$
3. for all $a, b, c \in M : (a \leq b \wedge b \leq c) \Rightarrow a \leq c$

We say that a tocm is a consistent totally ordered commutative monoid (ctocm), if for all $a, b, c \in M : a \leq b \Rightarrow a \oplus c \leq b \oplus c$

Important examples of commutative monoids are:

- $(N, +, 0)$, where N is the set of natural numbers, $+$ the usual addition, and 0 also as usual.
- $(N, \cdot, 1)$, with $\cdot$ the usual multiplication on N.
- $(Z, +, 0)$, with Z the set of whole numbers and $+$ the usual addition on Z.
- (({true,false},$\vee$,false), where $\vee$ is the usual or-operation.
- (({true,false},$\wedge$,false), where $\wedge$ is the usual and-operation.

Important examples of ctocm's are:

- $(N, +, 0, \leq)$, with $\leq$ the normal "lesser than or equal to" relation on N.
- $(N, +, 0, \geq)$, with $\geq$ the normal "greater than or equal to" relation on N.
- $(Z, +, 0, \leq)$
- $(Z, +, 0, \geq)$
- (({true,false},$\vee$,false, $\preceq$), where $\vee$ is the usual or-operation, and $\preceq$ is one of the 2 possible total orderings on {true,false}.
- (({true,false},$\wedge$,false, $\preceq$), where $\wedge$ is the usual and-operation, and $\preceq$ is one of the 2 possible total orderings on {true,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.

First we view a decision problem Π as a 3-tupple (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 2.6 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-tupple $D = (G_D, I_D)$, 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 $\Pi =$ HAMILTONIAN CIRCUIT (for undirected graphs). 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 all necessary information on the directions of the edges, in some coded form.

DEFINITION 2.7 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 $s_{\Pi|_\Theta} = s_\Pi|_{D_{\Pi|_\Theta}}$.

The above definition gives the natural way of restricting a graph problem to a class of graphs. Next we give a variant of the notion of polynomial transformation of decision problems (see e.g. [12], 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 2.8 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 $D \in D_1$, $s(f(D)) \leq p(s(D))$

The following result can be obtained in the same way as the similar results for (normal) polynomial transformations of decision problems.

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 Local condition composition problems and edge condition composition problems

In this section we define LCC, the class of local condition composition problems, and ECC, the class of the edge condition composition problems. Both are subclasses of the class of graph decision problems (and of NP). First we give the definition of basic local condition composition problems.

DEFINITION 2.9 Let $\Pi = (D_\Pi, Y_\Pi, s)$ be a graph decision problem. We say that Π is a *basic local condition problem*, if and only if there exist

- non-negative integers $m, c \in N$
- m commutative monoids $(M^1, \oplus^1), \ldots, (M^m, \oplus^m)$
- a ctocm $(M^{m+1}, \oplus^{m+1}, \preceq)$

such that

- each $D \in D_\Pi$ is of the form $(G, (X, Y, R_1, \ldots, R_m, K, I))$, where
  - G is an undirected graph
  - X is a finite set with $s(D) \geq |X|$
  - Y is a finite set with $s(D) \geq |Y|$
  - for all $i, 1 \leq i \leq m$, R_i denotes a subset of M^i
  - $K \in M^{m+1}$

- for all $i, 1 \leq i \leq m+1$, there exists a function val_i, that maps all 4-tupples, consisting of an instance $D = (G = (V,E),(X,Y,R_1,\ldots,R_m, K, I)) \in D_\Pi$, a vertex $v \in V$, and functions $f : N_c(v) \rightarrow X$, $g : M_c(v) \rightarrow Y$, to elements of M_i, such that for all (constants) $d \in N^+$:
  1. there exists an algorithm that calculates $val_i(D,v,f,g)$, for all $D = (G = (V,E),(X,R_1,\ldots,R_m,K,I)) \in D_\Pi$, $v \in V$, $f : N_c(v) \rightarrow X$, $g : M_c(v) \rightarrow Y$ with degree$(G) \leq d$, in time, polynomial in $s(D)$.
  2. if $1 \leq i \leq m$, there is a polynomial p_i, such that for all $D = (G = (V,E),(X,Y,R_1,\ldots,R_m,K,I)) \in D_\Pi$, with degree$(G) \leq d$ and subsets $W \subseteq V$: $|\{ \bigoplus^i_{w \in W} val_i(D,w,f|_{N_c(w)}, g|_{M_c(w)}) \mid f : N_c(W) \rightarrow (X), g : M_c(W) \rightarrow Y\}| \leq p_i(s(D))$.
  3. there exists an algorithm that calculates $a \oplus^i b$ for given a, b, such that there are $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, with degree$(G) \leq d$, $W_1 \subseteq V$, $W_2 \subseteq V$, $W_1 \cap W_2 = \emptyset$, $f : N_c(W_1 \cup W_2) \rightarrow X$, $g : M_c(W_1 \cup W_2) \rightarrow Y$, $a = \bigoplus^i_{w \in W_1} val_i(D,w,f|_{N_c(w)}, g|_{M_c(w)})$ and $b = \bigoplus^i_{w \in W_2} val_i(D,w,f|_{N_c(w)}, g|_{M_c(w)})$, in time, polynomial in $s(D)$.
  4. if $i = m+1$, then there exists an algorithm, that calculates whether $a \preceq b$ for given a, b, such that there are $D = (G = (V,E),(X,Y,R_1,\ldots,R_m,K,I)) \in D_\Pi$, with degree$(G) \leq d$, $W \subseteq V$, $f_1 : N_c(W) \rightarrow X$, $f_2 : N_c(W) \rightarrow X$, $g_1 : M_c(W) \rightarrow Y$, $g_2 : M_c(W) \rightarrow Y$, $a = \bigoplus^{m+1}_{w \in W} val_{m+1}(D,w,f_1|_{N_c(w)}, g_1|_{M_c(w)})$ and $b = \bigoplus^{m+1}_{w \in W} val_{m+1}(D,w,f_2|_{N_c(w)}, g_2|_{M_c(w)})$ or $b = K$, in time polynomial in $s(D)$.
  5. if $1 \leq i \leq m$, there exists an algorithm that calculates for all $D = (G = (V,E),(X,Y,R_1,\ldots,R_m,K,I)) \in D_\Pi$ with degree$(D) \leq d$, $f : V \rightarrow X$, $g : E \rightarrow Y$ and given $a = \bigoplus^i_{w \in V} val_i(D,w,f|_{N_c(w)}, g|_{M_c(w)})$ whether $a \in R_i$, in time polynomial in $s(D)$.
- For all $D = (G = (V,E),(X,Y,R_1\ldots,R_m,K,I)) \in D_\Pi : D \in Y_\Pi$, if and only if there exist functions $f : V \rightarrow X, g : E \rightarrow Y$, with

1. $\forall i, i \leq i \leq m : \bigoplus_{v \in V}^{i} val_i(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \in R_i$
2. $\bigoplus_{v \in V}^{m+1} val_{m+1}(D, v, f|_{N_c(v)}, g|_{M_c(v)}) \preceq K.$

The given definition of basic local condition composition problems may look at first sight very complicated, but in general it will be not very difficult to use. For a more intuitive introduction in the notion, see section 5.1.

Next we define the class of local condition composition problems, LCC.

DEFINITION 2.10 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.

DEFINITION 2.11 Let $\Pi = (D_\Pi, Y_\Pi, s)$ be a graph decision problem. We say that Π is a *basic edge condition problem*, if and only if there exist

- a non-negative integer $m \in N$
- m commutative monoids $(M^1, \oplus^1), \ldots, (M^m, \oplus^m)$
- a ctocm $(M^{m+1}, \oplus^{m+1}, \preceq)$

such that

- each $D \in D_\Pi$ is of the form $(G, (X, Y, R_1, \ldots, R_m, K, I))$, where
  - G is an undirected graph
  - X is a finite set with $s(D) \geq |X|$
  - Y is a finite set with $s(D) \geq |Y|$
  - for all $i, 1 \leq i \leq m$, R_i denotes a subset of M^i
  - $K \in M^{m+1}$

- for all $i, 1 \leq i \leq m+1$, there exists a function val_i, that maps all 4-tupples, consisting of an instance $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, a vertex $v \in V$, and functions $f : N_c(v) \rightarrow X$, $g : M_c(v) \rightarrow Y$, to elements of M_i, such that:
  1. there exists an algorithm that calculates $val_i(D, e, f, g)$, for all $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, $e \in E$, $f : N_c(e) \rightarrow X$, $g : M_c(e) \rightarrow Y$ in time, polynomial in $s(D)$.
  2. if $1 \leq i \leq m$, there is a polynomial p_i, such that for all $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, and subsets $E' \subseteq E$: $\{ \bigoplus^i_{w \in W} val_i(D, w, f|_{N_0(e)}, g|_{\{e\}}) | f : N_c(W) \rightarrow (X), g : M_c(W) \rightarrow Y \}| \leq p_i(s(D))$
  3. there exists an algorithm that calculates $a \oplus^i b$ for given a, b, such that there are $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, $E_1 \subseteq E$, $E_2 \subseteq E$, $E_1 \cap E_2 = \emptyset$, $f : N_c(E_1 \cup E_2) \rightarrow X$, $g : M_c(E_1 \cup E_2) \rightarrow Y$, $a = \bigoplus^i_{e \in E_1} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}})$ and $b = \bigoplus^i_{e \in E_2} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}})$, in time, polynomial in $s(D)$.
  4. if $i = m+1$, then there exists an algorithm, that calculates whether $a \preceq b$ for given a,b, such that there are $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, $E' \subseteq E$, $f_1 : N_c(E') \rightarrow X$, $f_2 : N_c(E') \rightarrow X$, $g_1 : M_c(E') \rightarrow Y$, $g_2 : M_c(E') \rightarrow Y$, $a = \bigoplus^{m+1}_{e \in E'} val_{m+1}(D, e, f_1|_{N_0(e)}, g_1|_{\{e\}})$ and $b = \bigoplus^{m+1}_{e \in E'} val_{m+1}(D, e, f_2|_{N_0(e)}, g_2|_{\{e\}})$ or $b = K$, in time polynomial in $s(D)$.
  5. if $1 \leq i \leq m$, there exists an algorithm that calculates for all $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$, $f : V \rightarrow X$, $g : E \rightarrow Y$ and given $a = \bigoplus^i_{e \in E} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}})$ whether $a \in R_i$, in time polynomial in $s(D)$.
- For all $D = (G = (V,E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi : D \in Y_\Pi$, if and only if there exist functions $f : V \rightarrow X$, $g : E \rightarrow Y$, with
  1. $\forall i, 1 \leq i \leq m : \bigoplus^i_{e \in E} val_i(D, v, f|_{N_0(e)}, g|_{\{e\}}) \in R_i$
  2. $\bigoplus^{m+1}_{e \in E} val_{m+1}(D, v, f|_{N_0(e)}, g|_{\{e\}}) \preceq K$.

DEFINITION 2.12 Let Π be a graph decision problem. We say that Π is a 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.

Lemma 2.4 *Let Π be a basic edge condition composition problem. Then $\Pi \in LCC$.*

Proof. Let m, $(M^1, \oplus^1)$,...,$(M^m, \oplus^m)$, $(M^{m+1}, \oplus^{m+1}, \preceq)$, val_1, ..., val_{m+1} be as indicated by the definition of basic ECC problem, applied to Π. To each graph $G = (V, E)$, appearing in one or more instances $D \in D_\Pi$, one can add an (arbitrary) assignation ch_G of each of the edges $e \in E$ to one of its adjacent vertices, i.e. $\forall (v, w) \in E : ch_G((v, w)) = v \vee ch_G((v, w)) = w$. We denote $ch_G^{-1}(v) = \{e \in E | ch_G(e) = v\}$. Note that $|ch_G^{-1}(v)| \leq degree(G)$.

Now we use a transformation from Π to a (very similar) other graph decision problem Π', by mapping each $D = (G = (V, E), (X, Y, R_1, \ldots, R_m, K, I)) \in D_\Pi$ to $\psi(D) = (G = (V, E), (X, Y, R_1, \ldots, R_m, K, (I, ch_G)))$, i.e. ch_G is included in the problem instances. Now Π' is a basic LCC-problem: take c=1, and let for all i, $1 \leq i \leq m$, $v \in V$:

$$val_i'(\psi(D), v, f|_{N_1(v)}, g|_{M_1(v)}) = \bigoplus_{e \in ch^{-1}(v)}^{i} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}}).$$

It is easy to see that the val_i'-functions can be calculated in polynomial time. Further we have that:

$$\begin{aligned} &\bigoplus_{v \in V}^{i} val_i'(\psi(D), v, f|_{N_1(v)}, g|_{M_1(v)}) = \\ &\bigoplus_{v \in V}^{i} \bigoplus_{e \in ch^{-1}(v)}^{i} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}}) = \\ &\qquad \bigoplus_{e \in E}^{i} val_i(D, e, f|_{N_0(e)}, g|_{\{e\}}). \end{aligned}$$

The remaining details are left to the reader. **Q.E.D.**

As an easy, but important corollary we have:

Theorem 2.5 *$ECC \subseteq LCC$.*

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

In this section we will 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. First we give some necessary lemma's.

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

Proof. We use induction to c. For $c = 0$ we have $v \in N_0(X_j) = X_j$, hence, by definition, $v \in X_k = N_0(X_k)$.

Suppose the lemma holds for all integers up to $c - 1$. There must be a vertex $w \in X_j$ with $d_G(v, w) \leq c$. If $d_G(v, w) \leq c - 1$, then $v \in N_{c-1}(X_j)$, and hence by induction, $v \in N_{c-1}(X_k) \subseteq N_c(X_k)$. So now suppose $d_G(v, w) = c$. Let x be the one-but-last vertex on the path in G from v to w, i.e. $d_G(v, x) = c - 1$ and $(x, w) \in E$. There must be an $i' \in I$ with $x \in X_{i'}$, $w \in X_{i'}$, by definition. Note that k is on the path from i to i' in T or k is on the path from i' to j. We consider both cases.

Case 1: k is on the path from i to i'. Note that $w \in X_{i'} \Rightarrow v \in N_{c-1}(X_{i'})$. By induction it follows that $v \in N_{c-1}(X_k) \subseteq N_c(X_k)$.

Case 2: k is on the path from i' to j. From $w \in X_{i'}$ and $w \in X_j$ it follows that $w \in X_k$. Hence $v \in N_c(X_k)$.

Q.E.D.

Lemma 3.2 *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. $M_c(X_i) \cap M_c(X_i) \subseteq M_c(X_k)$

Proof.

1. Suppose $v \in N_c(X_i) \cap N_c(X_j)$. There must be some $i' \in I$ with $v \in X_{i'}$. Now k is on the path from i to i' or k is on the path from i' to j. In both cases it follows from lemma 3.1 that $v \in N_c(X_k)$. Hence we have $N_c(X_i) \cap N_c(X_j) \subseteq N_c(X_k)$.

2. $(v,w) \in M_c(X_i) \cap M_c(X_j) \Rightarrow v \in N_c(X_i) \cap N_c(X_j) \wedge w \in N_c(X_i) \cap N_c(X_j) \Rightarrow v \in N_c(X_k) \wedge w \in N_c(X_k) \Rightarrow (v,w) \in M_c(X_k)$.

Q.E.D.

Theorem 3.3 *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, $(M^1, \oplus^1)$, ..., $(M^m, \oplus^m)$, $(M^{m+1}, \oplus^{M+1}, \preceq)$, val_1, ..., 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 $G \in \mathrm{TWD}(k,d)$, 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 $i \in I$, (i.e. the direct descendants of i in the rooted tree T, is denoted by $\mathrm{sons}(i)$; the father of i in the tree (if $i \neq$ root), is denoted by $\mathrm{father}(i)$, and the set of all descendants of i, including i self, is denoted by $\mathrm{dec}(i)$. Further for each $i \in I$, we pose a (total) ordering upon the sons of i.

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

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

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

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

Lemma 3.3.1 *For all $v \in V$ there is a unique $i \in I$, with $v \in Y_i$.*

Proof. Let $v \in V$. By definition, there is an $i \in I$, with $v \in X_i$. Now either $v \in X_{\text{root}}$, or there is a node $j \in I$, on the path from i to root with $v \in X_j$ and $v \notin X_{\text{father}(j)}$. Hence $\exists i \in I$ with $v \in Y_i$.

Next suppose we have $v \in Y_i \cap Y_j$, $i \neq j$. Note that we have $v \in X_i \cap X_j$. Now note that either father(i) or father(j) exists and is on the path from i to j, and hence $v \in X_{\text{father}(i)}$ or $v \in X_{\text{father}(j)}$. This contradicts $v \in Y_i \cap Y_j$. **Q.E.D.**

The following two lemmas are easily verified.

Lemma 3.3.2 *For all $i \in I$:* $Z_i = \left(\bigcup_{j \in \text{sons}(i)} Z_j \cup Y_i \right)$.

Lemma 3.3.3 *For all $i \in I$:* $W_i = \left(\bigcup_{j \in \text{sons}(i)} W_j \cup X_i \right)$.

Lemma 3.3.4 *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. $M_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \cap M_c(W_{j_\alpha}) \subseteq M_c(X_i) \cap M_c(X_{j_\alpha})$.

Proof. 1. Suppose $v \in N_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \cap N_c(W_{j_\alpha})$. There must be some $j \in \text{dec}(j_\alpha)$, with $v \in N_c(X_j)$. We consider two cases.

Case 1: $v \in N_c(X_i)$. It follows that j_α is on the path from i to j in T, hence, by lemma 3.2, $v \in N_c(X_{j_\alpha})$.

Case 2: $\exists \beta, 1 \leq \beta \leq \alpha - 1 : v \in N_c(W_{j_\beta})$. There must be some $j' \in \text{dec}(j_\beta)$, with $v \in N_c(X_{j'})$. Now i and j_α are on the path from j to j' in T, hence, by lemma 3.2, $v \in N_c(X_i)$ and $v \in N_c(X_{j_\alpha})$.

2. Similar. **Q.E.D.**

In the remainder, we will use the expression $\sigma_p(v, f, g)$ 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 $\bigoplus^p_{v \in S} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$, 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 will now calculate for each node $i \in I$ a table, called TABLE(i), which contains $(m+3)$-tupples 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

$$(\tilde{f} : N_c(X_i) \to X,\ \tilde{g} : M_c(X_i) \to Y,\ r_1, \ldots, r_{m+1}) \in \text{TABLE}(i) \Leftrightarrow \quad (1)$$

$$\begin{aligned}
&\exists f : N_c(W_i) \to X,\ g : M_c(W_i) \to Y, \text{ with} \\
&\quad f|_{N_c(X_i)} = \tilde{f}, \\
&\quad g|_{M_c(X_i)} = \tilde{g}, \\
&\quad \forall p,\ 1 \le p \le m+1,\ \sigma_p(Z_i, f, g) = r_p \\
&\text{and } \forall f : N_c(W_i) \to X,\ g : M_c(W_i) \to Y, \text{ with} \\
&\quad f|_{N_c(X_i)} = \tilde{f}, \\
&\quad g|_{M_c(X_i)} = \tilde{g}, \\
&\quad \forall p,\ 1 \le p \le m,\ \sigma_p(Z_i, f, g) = r_p \\
&\quad \text{one has} : r_{m+1} \prec \sigma_{m+1}(Z_i, f, g).
\end{aligned}$$

Note that for each $\tilde{f} : N_c(X_i) \to X, \tilde{g} : M_c(X_i) \to Y, r_1 \in M^1, \ldots, r_m \in M^m$, there will be at most one $r_{m+1} \in M^{m+1}$, with $(\tilde{f}, \tilde{g}, r_1, \ldots, r_{m+1}) \in$ 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, $1 \le p \le m$.

The algorithm starts by first recursively calculating the TABLE's for all sons of i, and then calculating temporary tables TEMP(i, α), with $0 \le \alpha \le r$, 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):

$$(\tilde{f} : N_c(X_i) \to X,\ \tilde{g} : M_c(X_i) \to Y,\ r_1, \ldots, r_{m+1}) \in \text{TEMP}(i, \alpha) \Leftrightarrow \quad (2)$$

$$\begin{aligned}
&\exists f : N_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \to X, \\
&\qquad g : M_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \to Y, \text{ with} \\
&\quad f|_{N_c(X_i)} = \tilde{f}, \\
&\quad g|_{M_c(X_i)} = \tilde{g}, \\
&\quad \forall p,\ 1 \le p \le m+1 :\ \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p \\
&\text{and } \forall f : N_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \to X, \\
&\qquad g : M_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \to Y, \text{ with}
\end{aligned}$$

$$f|_{N_c(X_i)} = \tilde{f},$$
$$g|_{M_c(X_i)} = \tilde{g},$$
$$\forall p,\ 1 \leq p \leq m:\ \sigma_p(Z_{j-1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p$$
$$\text{one has}: r_{m+1} \prec \sigma_{m+1}(Z_{j-1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g).$$

Note that it follows from lemma's 3.3.2 and 3.3.3, that equation (1) follows from equation (2), with $\alpha = r$ (= the number of sons of i).

We now claim that the following pair of subroutines will calculate the TABLEs and TEMPs correctly. Basically, the algorithm works as follows: first we calculate (recursively) TABLE(j_α), for all sons j_α of i. Then we calculate 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 calculate the tables TEMP(i, α), by composing the tables TEMP($i, \alpha - 1$) and TABLE(j_α). From the above observation it follows that 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 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 3.3.5 *After execution of CALCULATE_TABLE(i) equation (1) holds and after execution of CALCULATE_TEMP(i, α) equation (2) holds.*

Proof. Consider $i \in I$, let $|\text{sons}(i)| = r$, and let $0 \leq \alpha \leq r$. For $\alpha = 0$, it can be verified directly, that equation (2) holds after execution of CALCULATE_TEMP$(i, 0)$. So let $\alpha \geq 1$. By using induction, one may assume that the equations hold for TABLE(j_α) and for TEMP$(i, \alpha - 1)$.

Now we first suppose that after execution of TEMP(i, α), $(f, g, r_1, \ldots, r_{m+1}) \in$ TEMP(i, α). It follows from the algorithm that there must be $(f, g, s_1, \ldots, s_{m+1}) \in$ TEMP$(i, \alpha - 1)$, and $(f', g', t_1, \ldots, t_{m+1}) \in$ TABLE(j_α), with: $f' : N_c(X_{j_\alpha}) \rightarrow X$, $g' : M_c(X_{j_\alpha lpha}) \rightarrow Y$, and $\forall v \in N_c(X_i) \cap N_c(X_{j_\alpha})$: $f(v) = f'(v)$, and $\forall e \in M_c(X_i) \cap M_c(X_{j_\alpha})$: $g(e) = g'(e)$, and $\forall p, 1 \leq p \leq m+1$: $r_p = s_p \oplus t_p$. By induction, there are $\tilde{f} : N_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \rightarrow X$,

$\tilde{g} : M_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \rightarrow Y$, with $\tilde{f}|_{N_c(X_i)} = f$; $\tilde{g}|_{M_c(X_i)} = g$; $\forall p,\ 1 \leq p \leq m+1$: $\sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_{\alpha-1}} \cup Y_i, \tilde{f}, \tilde{g}) = r_p$; and there are $\tilde{f}' : N_c(W_{j_\alpha}) \rightarrow X$, $\tilde{g}' : M_c(W_{j_\alpha}) \rightarrow Y$, with $\tilde{f}'|_{N_c(X_{j_\alpha})} = f'$; $\tilde{g}'|_{M_c(X_{j_\alpha})} = g'$; $\forall p,\ 1 \leq p \leq m+1$: $\sigma_p(Z_{j_\alpha}, \tilde{f}', \tilde{g}') = s_p$;

Now we define functions $F : N_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \rightarrow X$, $G : M_c(W_{j_1} \cup \ldots \cup W_{j_\alpha} \cup X_i) \rightarrow Y$, as follows.

$$F(v) = \begin{cases} \tilde{f}(v), & \text{if } v \in N_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \\ \tilde{f}'(v), & \text{if } v \in N_c(W_{j_\alpha}) \end{cases}$$

$$G(e) = \begin{cases} \tilde{g}(e), & \text{if } e \in M_c(W_{j_1} \cup \ldots \cup W_{j_{\alpha-1}} \cup X_i) \\ \tilde{g}'(e), & \text{if } e \in M_c(W_{j_\alpha}) \end{cases}$$

It follows from lemma 3.3.4, that the definition of F and G is correct. Now, for all p, $1 \leq p \leq m+1$, one has $\sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, F, G) = \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_{\alpha-1}} \cup Y_i, F, G) \oplus \sigma_p(Z_{j_\alpha}, F, G) = s_p \oplus r_p$. So now we have proven that the following holds.

$$(\tilde{f} : N_c(X_i) \rightarrow X,\ \tilde{g} : M_c(X_i) \rightarrow Y,\ r_1, \ldots, r_{m+1}) \in \mathrm{TEMP}(i, \alpha) \Rightarrow \quad (3)$$

$$\begin{aligned} &\exists f : N_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \rightarrow X, \\ &\qquad g : M_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \rightarrow Y, \text{ with} \\ &f|_{N_c(X_i)} = \tilde{f}, \\ &g|_{M_c(X_i)} = \tilde{g}, \\ &\forall p,\ 1 \leq p \leq m+1 :\ \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p. \end{aligned}$$

Next suppose we have $\tilde{f} : N_c(X_i) \rightarrow X$, $\tilde{g} : M_c(X_i) \rightarrow$, $r_1 \in M^1$, $\ldots$, $r_{m+1} \in M^{m+1}$, such that

$$\begin{aligned} &\exists f : N_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \rightarrow X, \\ &\qquad g : M_c(W_{j_1} \cup \ldots W_{j_\alpha} \cup X_i) \rightarrow Y, \text{ with} \\ &f|_{N_c(X_i)} = \tilde{f}, \\ &g|_{M_c(X_i)} = \tilde{g}, \\ &\forall p,\ 1 \leq p \leq m+1 :\ \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_\alpha} \cup Y_i, f, g) = r_p. \end{aligned}$$

We claim that now $\exists t \preceq r_{m+1}$, with $(\tilde{f}, \tilde{g}, r_1, \ldots, r_m, t) \in \mathrm{TEMP}(i, \alpha)$.

Write $f' = f|_{N_c(W_{j_1} \cup \ldots W_{j_{\alpha-1}} \cup X_i)}$, $g' = g|_{M_c(W_{j_1} \cup \ldots W_{j_{\alpha-1}} \cup X_i)}$, $f'' = f|_{N_c(W_{j_\alpha})}$, $g'' = g|_{M_c(W_{j_\alpha})}$, $\forall p,\ 1 \leq p \leq m+1$: $r'_p = \sigma_p(Z_{j_1} \cup \ldots \cup Z_{j_{\alpha-1}} \cup Y_i, f', g')$, and $r''_p = \sigma_p(Z_{j_\alpha}, f, g)$.

By induction, there are $t' \preceq r'_{m+1}$ and $t'' \preceq r''m+1$, such that $(f'|_{N_c(X_i)}, g'|_{M_c(X_i)}, r'_1, \ldots, r'_m, t') \in \mathrm{TEMP}(i, \alpha - 1)$, and $(f''|_{N_c(X_{j_\alpha})}, g''|_{M_c(X_{j_\alpha})}, r''_1, \ldots, r''_m, t'') \in \mathrm{TABLE}(i)$. From the algorithm it now follows that table $\mathrm{TEMP}(i, \alpha)$ will contain the element: $(\tilde{f}, \tilde{g}, r'_1 \oplus^1 r''_1, \ldots, r'_m \oplus^m r''_m, t' \oplus^{m+1} t'')$. By noting that $r'_p \oplus^p r''_p = r_p$ and $t' \oplus^{m+1} t'' \preceq r'_{m+1} \oplus^{m+1} r''_{m+1} = r_{m+1}$, the claim follows.

Finally we note that for each $f : N_c(X_i) \to X$, $g : M_c(X_i) \to Y$, $r_1 \in M^1, \ldots, r_m \in M^m$, there is at most *one* $r_{m+1} \in M^{m+1}$, with $(f, g, r_1, \ldots, r_{m+1}) \in \mathrm{TEMP}\ (i, \alpha)$.

Claim 3.3.5 now follows from the above 3 observations. **Q.E.D.**

Next we show that is is easy 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 3.3.6 *$D \in Y_\Pi$, if and only if there are $r_1 \in R_1$, $r_2 \in R_2$, $\ldots$, $r_m \in R_m$, $r_{m+1} \in M^{m+1}$, with $r_{m+1} \prec K$, and $f : N_c(X_{\mathrm{root}}) \to X$, $g : M_c(X_{\mathrm{root}}) \to X$, such that $(f, g, r_1, \ldots, r_m, r_{m+1}) \in$ TABLE(root).*

Proof. First we note that $W_{\mathrm{root}} = Z_{\mathrm{root}} = V$.

Now suppose $D \in Y_\Pi$. By definition, there are $f : V \to X$, $g : E \to Y$, with $\forall p, 1 \leq p \leq m : \sigma_p(V, f, g) \in R_p$ and $\sigma_{m+1}(V, f, g) \prec K$. Denote $s_p = \sigma_p(V, f, g)$ $(1 \leq p \leq m)$ and $\tilde{f} = f|_{N_c(X_{\mathrm{root}})}$, $\tilde{g} = g|_{M_c(X_{\mathrm{root}})}$. It follows from claim 3.3.5 that either $(\tilde{f}, \tilde{g}, s_1, \ldots, s_{m+1}) \in$ TABLE(root), or there is a $r_{m+1} \prec s_{m+1} \prec K$, and $(\tilde{f}, \tilde{g}, s_1, \ldots, r_{m+1}) \in$ TABLE(root).

Next suppose there are $r_1 \in R_1$, $r_2 \in R_2$, $\ldots r_m \in R_m$, $r_{m+1} \in M^{m+1}$, with $r_{m+1} \prec K$, and $f : N_c(X_{\mathrm{root}}) \to X$, $g : M_c(X_{\mathrm{root}}) \to X$, such that $(f, g, r_1, \ldots, r_m, r_{m+1}) \in$ TABLE(root). Hence there are $\tilde{f} : V \to X$, and $\tilde{g} : E \to Y$), with $\forall p, 1 \leq p \leq m + 1$): $\sigma_p(V, f, g) = r_p$. It follows that $D \in Y_\Pi$. **Q.E.D.**

From claim 3.3.6 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 $\mathrm{TEMP}(i, \alpha)$ (and consequently, of $\mathrm{TABLE}(i)$), 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 $(f, g, r_1, \ldots, r_{m+1}) \in \text{TEMP}(i, \alpha)$, 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 $\text{TEMP}(i, \alpha)$ is bounded by

$$|X|^{c_1} \cdot |Y|^{c_2} \cdot \prod_{p=1}^{m} p_p(s(D)),$$

which is a polynomial in $s(D)$.

It follows that 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 calculate 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 $(f, g, r_1, \ldots, r_m, r_{m+1}) \in \text{TABLE(root)}$, whether $r_1 \in R_1, \ldots, r_m \in R_m$ and $r_{m+1} \prec K$. As TABLE(root) is of polynomial size, this last step of the algorithm also uses polynomial time. **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 and $\frac{1}{2}k(k+1)$ respectively, i.e. these sizes are bounded by a constant, even if there is no bound on the degree of the graphs. In this way we obtain the following result:

Theorem 3.4 *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 3.5 *(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 3.3 and theorem 3.4. **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). One can modify the definition of basic LCC problem to that of basic C local condition composition problems, and the definition of basic ECC problem to that of basic C edge condition composition problem by adding the following 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. Conditions 1, 3, 4 and 5 of the definition still hold when we replace the sentence "time, polynomial in $s(D)$" by "constant time".
3. The degree of the polynomial $\prod_{p=1}^{k} p_p$ is at most $C - 1$.

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 will 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 $\Pi \in C\text{-ECC}$, *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* $\leq k$.

The following relations between the classes C-LCC, C-ECC, LCC and ECC can be obtained without much difficulty.

Theorem 4.2 *(i)* $C\text{-ECC} \subseteq C\text{-LCC}$.
(ii) $C\text{-LCC} \subseteq LCC$.
(iii) $C\text{-ECC} \subseteq ECC$.

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, assuming that the tree-decompositions with the required tree-width are given. It is presently unknown whether one can find tree-decompositions with logarithmic treewidth in polynomial time.

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

(ii) Let $\Pi \in C\text{-ECC}$, *and let* $k \in N^+$. *Let* Θ *be a class of graphs, with* $\exists c > 0$: $\forall G \in \Theta \Rightarrow$ *treewidth*$(G) \leq k$. *Then there exists a polynomial 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* $\leq c \cdot \log(|V|)$.

Proof. (i) Use the same algorithm as in the case of the constant bounded treewidth. We now will estimate the size of the tables TEMP and TABLE. Note that the size of $N_c(v)$, or $M_c(v)$, $v \in V$, is bounded by a constant, that only depends on k, d and c. Hence there are constants c_1, c_2, such that $|N_c(X_i)| \leq c_1 \cdot \log(|V|)$ and $|M_c(X_i)| \leq c_2 \cdot \log(|V|)$, for all $G = (V, E) \in \Theta$, and graph-decompositions $(\{X_i \mid i \in I\}, T)$ of G with treewidth $\leq c \cdot \log(|V|)$. It follows that the size of a table TEMP(i, α) (or TABLE(i)), is bounded by

$$|X|^{c_1 \cdot \log(|V|)} \cdot |Y|^{c_2 \cdot \log(|V|)} \cdot \prod_{p=1}^{m} p_p(s(D)).$$

Using that $s(D) \geq |V|$, and that $|X|$ and $|Y|$ are bounded by a constant, it follows that the sizes of the TEMPs and TABLEs are bounded by a polynomial in $s(D)$. The remainer of the proof is similar to the proof of theorem 3.3.

(ii) Similar. **Q.E.D.**

5 Problems in LCC and ECC

In this section we will show for a large number of NP-complete graph decision problems, that they are in LCC or in ECC. First in section 5.1 we will give some of the basic techniques we use to prove problems to be in (C-)LCC and (C-)ECC, and give some intuitive ideas behind these notions. In section 5.2 we give a give for many NP-complete graph decision problems (an indication of the) proof that they are in LCC, ECC, C-LCC or C-ECC. For many problems, the same techniques will be used to transform them to a basic LCC or ECC problem. We have restricted ourselves to a number of graph decision problems, appearing in [12]. For full descriptions of the problems, the reader is also referred to this reference. We omitted most details of proofs, in this section. Most omitted details are easy to obtain.

5.1 Some basic techniques

First we give a more intuitive idea of what a (basic) LCC or ECC problem is. A basic LCC-problem is a problem of the following type (with some extra restrictions):

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

The extra restrictions basically say, that where necessary, operations $\oplus$, $\in R_p$, and $\prec$ can be done in polynomial time; the functions val_p can be

calculated in polynomial time, and, for $p \neq m+1$, the number of different values that $\bigoplus^p_{v \in S} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ can assume, with some fixed $S \subseteq V$, over all possible f and g, is polynomially bounded. A basic ECC-problem has the following type (with similar restrictions):

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

- $\forall p, 1 \leq p \leq m : \bigoplus^p_{e \in E} val_p(D, e, f|_{N_0(e)}, g|_{\{e\}}) \in R_p$
- $\bigoplus^{m+1}_{e \in E} val_{m+1}(D, e, f|_{N_c(e)}, g|_{\{e\}}) \preceq K$

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

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 $((\{\underline{\text{true}}, \underline{\text{false}}\}, \wedge, \underline{\text{true}})$, where $\wedge$ is the usual and-operation. 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 $(\{\underline{\text{true}}, \underline{\text{false}}\}, \vee, \underline{\text{false}})$, where $\vee$ is the usual or-operation.

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 $s(D)$.

If we know some fixed upperbound, say L on the maximal value of these sums (for instances with a 'yes'-answer), we can use the commutative monoid with elements $\{0, 1, \ldots, L, L+1\}$, and addition $\oplus$ on this set, with $i \oplus j = i + j$, if $i + j \leq L + 1$, and $i \oplus j = L + 1$, if $i + j > L + 1$. In this way the number of different values the sums can attain is bounded by a constant, which is useful, when we want to prove membership in C-LCC or C-ECC, with C as small as possible.

Conditions like $\bigoplus_{e \in E}^{p} val_p(D, e, f|_{N_0(e)}, g|_{\{e\}})$ can be expressed, while writing the problem as a basic LCC-problem, similar as in the proof of lemma 2.4.

With a similar technique, 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 a mapping $V \to E$, where each vertex v is mapped upon a neighboring edge e. (We have to assume that G does not have isolated vertices.))

5.2 A list of problems in LCC and ECC

5.2.1 Vertex cover [GT 1]

In [3] it was shown that the problem can be solved in linear times, for graphs with treewidth, bounded by some constant number k. This can also be shown with the following result.

Theorem 5.1 VERTEX COVER $\in$ 1-*ECC*.

Proof. There is a linear time transformation of VERTEX COVER to the following problem:

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

1. $\forall (v, w) \in E : f(v) = 1$ or $f(w) = 1$.
2. $\sum_{v \in V} f(v) \leq K$.

It is straightforward to see that this problem is in 1-ECC. **Q.E.D.**

5.2.2 Dominating set [GT 2]

Similar to VERTEX COVER, a linear time algorithm for this problem for graphs with constant treewidth, was given by Arnborg and Proskurowski in [3]. One also has:

Theorem 5.2 DOMINATING SET $\in$ *ECC.*

Proof. There is a linear time transformation from DOMINATING SET to the following problem:

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

1. $\forall (v, w) \in E : f(v) = w \Rightarrow f(w) = 1$.
2. $\forall (v, w) \in E : f(w) = v \Rightarrow f(v) = 1$.
3. $\forall v \in V : f(v) = 1$ or $f(v)$ is a neighbor of v.
4. The number of $v \in V$, with $f(v) = 1$, is at most K.

(The vertices v, with $f(v) = 1$ represent the set V'. For all other vertices v, $f(v)$ represents the neighbor of v, that is in V'.) The latter problem can easily be transformed to a basic ECC-problem, with standard techniques (see section 5.1.) **Q.E.D.**

5.2.3 Domatic Number [GT 3]

Scheffler and Seese [24] showed that DOMATIC NUMBER can be solved in linear time for graphs with given treedecomposition with constant bounded treewidth, and constant bounded degree. (To be precise, they prove that for constant k, d, K, the problem whether a given graph with treewidth at most k, degree at most d, has a domatic number that is at least K, is solvable in linear time (not calculating the time needed to find the tree-decomposition with the required treewidth). However, as the domatic number of a graph is at most its degree $+1$, it follows directly, that the general DOMATIC NUMBER problem (i.e. K is a part of the problem instance), also is solvable in linear time, for this class of graphs.) A similar result can also be obtained with the following theorem.

Theorem 5.3 DOMATIC NUMBER $\in$ 1-*LCC.*

Proof. We use that the domatic number of a graph is at most its degree $+1$. So we may assume that $K \leq d+1$, i.e. K is bounded by a constant. Further note that DOMATIC NUMBER has a linear transformation to the following problem:

> INSTANCE: Graph $G=(V,E)$, sets $X=\{1,\ldots,K\}$, $Y=\{0\}$.
> QUESTION: Are there functions $f: V \to X$, $g: E \to Y$, such that $\forall v \in V : (\forall i, 1 \leq i \leq K : \exists w \in N_1(v) : f(w)=i)$?

The latter problem can easily be transformed to a basic 1-LCC-problem. **Q.E.D.**

A result for graphs with no bound on the degree of the vertices, can be obtained by observing that the domatic number of a graph is at most the smallest degree of a vertex $+1$. As each graph with treewidth $\leq k$ has a vertex with degree $\leq k$ (use e.g. the characterization as partial k-tree), it follows that we may assume that we can bound K by a constant. Together with the following result, one obtains polynomial time algorithms for graphs with given tree-decomposition with constant bounded tree-width.

Theorem 5.4 DOMATIC NUMBER *for constant* $K \in$ *ECC.*

Proof. Note the equivalence to the following problem:

> INSTANCE: Graph $G=(V,E)$, sets $X=\{1,\ldots,K\} * V^K$, $Y=\{0\}$.
> QUESTION: Are there functions $f: V \to X$, $g: E \to Y$, such that
>
> 1. $\forall v \in V : \forall i, 2 \leq i \leq K+1 : f_i(v) \in N_1(v)$.
> 2. $\forall v \in V : \forall i, 2 \leq i \leq K+1$: if $f_i(v)=v$, then $f_1(v)=i-1$.
> 3. $\forall (v,w) \in E : \forall i, 2 \leq i \leq K+1$: if $f_i(v)=w$, then $f_1(w)=i-1$.
> 4. $\forall (v,w) \in E : \forall i, 2 \leq i \leq K+1$: if $f_i(w)=v$, then $f_1(v)=i-1$.

Here $f_2(v),\ldots,f_{K+1}(v)$, denote the vertices in $N_1(v)$, that are in the dominating sets $V_1,\ldots,V_K$; $V_i=\{v \in V \mid f_1(v)=i\}$. Now the problem is easily seen to be in ECC. **Q.E.D.**

This proof shows a technique, which often will be used to transform an (C-)LCC problem to an ECC-problem. Suppose we have a condition of the form $\forall v \in V : \exists w \in N_1(v) : Q(f(v), f(w), g((v,w)))$, where Q is some relation. Now replace the set X (of the old problem), by $X' = X * V$; add a condition that $f_2(v) \in N_1(v)$ for all $v \in V$; and require now, that for each edge $e = (v,w) \in E$: if $f_2(v) = w$, then $Q(f_1(v), f_1(w), g((v,w))$ and if $f_2(w) = v$, then $Q(f_1(w), f_1(v), g((v,w))$. Also require that for all $v \in V$: if $f_2(v) = v$, then $Q(f_1(v), f_1(v), g((v,w))$. If we require w to be a neighbor of v, instead of an element of $N_1(v)$, then we drop the last condition. For each $v \in V$, $f_2(v)$ is the edge leading to the neighbor w, such that $Q(f(v), f(w), g((v,w))$.

5.2.4 Chromatic Number [GT 4]

Arnborg and Proskurowski [3] showed that CHROMATIC NUMBER can be solved in linear time for graphs with constant bounded treewidth (and with given corresponding characterization as subgraph of a k-tree). By using that a graph with treewidth k always is $(k+1)$-colorable (this fact can easily be derived with help of the recursive definition of k-trees), one obtains the following result without difficulty:

Theorem 5.5 CHROMATIC NUMBER $\in$ 1-*ECC*.

5.2.5 Monochromatic triangle [GT 6]

Scheffler and Seese [24] proved this problem to be linear time solvable for graphs with constant bounded treewidth and degree. This result can also be obtained with the following theorem.

Theorem 5.6 MONOCHROMATIC TRIANGLE $\in$ 1-*LCC*.

Proof. Note that MONOCHROMATIC TRIANGLE has a graph-invariant linear transformation to the following problem:

> INSTANCE: Graph $G = (V,E)$, sets $X = \{0\}$, $Y = \{1,2\}$.
> QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that $\forall u \in V$: $\neg(\exists v,w \in N_1(v)$: $(v,w) \in E \wedge f((v,w)) = f((u,v)) = f((u,w))$.

The latter is easily transformed to a basic 1-LCC problem. **Q.E.D.**

If we replace "triangle" by any larger fixed complete subgraph, then one shows with a similar proof, that the resulting variant again is in 1-LCC.

We remark that there exist also linear time algorithms for the case that one has no bound on the degree of the graphs. By using lemma 5.23, it follows that for each triangle (u, v), (v, w), (w, u) in G, there must be some $i \in I$, with $u, v, w \in X_i$. (Similar for larger complete subgraphs of G.) This fact enables us to modify the basic algorithms of section 3, in order to obtain a linear time algorithm for MONOCHROMATIC TRIANGLE, or one of its variants with larger complete subgraphs, on graphs with given treedecompostion with constant bounded treewidth.

5.2.6 Feedback vertex set [GT 7]

Feedback vertex set is the first example in this list, of a problem, dealing with directed graphs. A directed graph can be seen as an undirected graph, with each edge labeled by it direction(s). (So there are 3 different labelings possible for each edge.) Hence, there is no real difference in the way undirected and directed graphs can be handled in this theory.

Theorem 5.7 FEEDBACK VERTEX SET $\in$ *ECC.*

Proof. First we may assume without loss of generality, that we have no isolated vertices in $G = (V, A)$. Note that the condition that V' contains at least one vertex from every directed cycle in G is equivalent to the condition that the subgraph of G, induced by $V - V'$ is cyclefree. Hence we can transform FEEDBACK VERTEX SET to the following problem:

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

1. $\forall (u, v) \in A$: $f(u) = \infty$ or $f(v) = \infty$ or $f(u) < f(v)$.
2. the number of $v \in V$ with $f(v) = \infty$ is at most K.

(The vertices v with $f(v) = \infty$ represent the set V'.) **Q.E.D.**

5.2.7 Feedback arc set [GT 8]

This problem can be handled, similar to FEEDBACK VERTEX SET.

Theorem 5.8 FEEDBACK ARC SET $\in$ *ECC.*

Proof. Transform to the following problem.

INSTANCE: Directed graph $G = (V, A)$, sets $X = \{0, 1, \ldots, |V| - 1\}$, $Y = \{0, 1\}$, positive integer $K \leq |V|$.
QUESTION: Are there functions $f : V \rightarrow X$, $g : A \rightarrow Y$, such that

1. $\forall e = (u, v) \in A$: $g(e) = 1$ or $f(u) < f(v)$.
2. $\sum_{e \in E} g(e) \leq K$.

(The edges e with $g(e) = 1$ represent the set A'.) Note that we can express the function $g : A \rightarrow Y$ as a function g' from the set of undirected edges E, obtained by ignoring the direction of the edges in A, to $(Y * Y)$. **Q.E.D.**

5.2.8 Partial feedback edge set [GT 9]

Scheffler and Seese [24] have shown that the problem with fixed maximum-circuit length L is solvable in linear time for graphs with given tree-decomposition with treewidth $\leq k$ and degree at most d, k and d fixed. One can show that the problem with fixed L, is in 1-LCC, thus obtaining the same result. The problem is open for graphs with arbitrary degree, and for the variant where L is variable.

Theorem 5.9 *For all* $L \in N^+, L \geq 3$, PARTIAL FEEDBACK EDGE SET *with maximum circuit length* $L \in$ 1-*LCC.*

Proof. Transform to the following problem.

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

1. $\forall v \in V$: There is no circuit in $N_L(v)$, with for every edge e in the circuit $f(e) = 0$.
2. $\sum_{e \in E} f(e) \leq K$.

Q.E.D.

5.2.9 Minimum maximal matching [GT 10]

Scheffler and Seese proved that the problem, whether the minimum maximal matching of a given graph G has at most K edges, (for fixed K), is solvable in linear time, for graphs with constant bounded treewidth and degree, given with the corresponding tree-decomposition. The following result shows that a similar result still holds, if we do not fix K, but let it be a part of the problem-instance.

Theorem 5.10 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.**

Also we have the following result.

Theorem 5.11 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.

5.2.10 Partition into triangles [GT 11]

Theorem 5.12 PARTITION INTO TRIANGLES $\in$ 1-*LCC*.

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

INSTANCE: Graph $G = (V, E)$, sets $X = \{0\}$, $Y = \{0, 1\}$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that $\forall u \in V$: $\exists e = (u, v) \in M_1(v)$, $e' = (u, w) \in M_1(v)$, with $f(e) = f(e') = 1$, and $(v, w) \in E$, and $f((v, w)) = 1$, and for all other edges $(v, x) \in E$: $f((v, x)) = 0$.

The latter problem is easily transformed to a basic 1-LCC problem. **Q.E.D.**

Theorem 5.13 PARTITION INTO TRIANGLES $\in$ *ECC*.

Proof. Use the technique, outlined in section 5.2.3. **Q.E.D.**

5.2.11 Partition into Isomorphic subgraphs [GT 12]

We will consider the subproblem of this problem, where we require H to be connected.

Theorem 5.14 PARTITION INTO ISOMORPHIC CONNECTED SUBGRAPHS $\in$ *LCC*.

Proof. We choose some arbitrary vertex $w \in V_H$. Each set V_i will be characterized by the vertex $v \in V_i$, that is mapped to w. We use a transformation to the following problem.

INSTANCE: Graph $G = (V_G, E_G)$, sets $X = V_G * V_H$, $Y = \{0\}$, graph $H = (V_H, E_H)$, vertex $w \in V_H$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: $f_2(v) = w \Leftrightarrow f_1(v) = v$.
2. $\forall v \in V$ for every $w' \in V_H$ that is adjacent in H to $f_2(v)$, there is a unique $v' \in V$, that is adjacent to v (in G), with $f(v') = (f_1(v), w')$.
3. $\forall (v, w) \in E$: if $f_1(v) = f_1(w)$, then $(f_2(v), f_2(w)) \in E_H$.

We leave the remainder of the proof to the reader. **Q.E.D.**

In a similar manner, one can handle the case that H has a fixed number of connected components. (One can choose a vertex w_i from every connected component H_i; by counting the number of times that $f_2(v) = w_i$, for each i, one can verify that each component of H appears often as a subgraph.) We also remark that if we fix H, then the problem is in 1-LCC. This follows by the observation, that for each vertex v, one can require that $d_G(v, f_1(v)) \leq |V_H|$. As $|V_H|$ is fixed, it follows that, for graphs with bounded degree, one can obtain a set X with fixed size.

5.2.12 Partition into Hamiltonian Subgraphs [GT 13]

Theorem 5.15 PARTITION INTO HAMILTONIAN SUBGRAPHS $\in$ 1-*LCC*.

Proof. Use the equivalence of the problem with the following problem:

INSTANCE: Directed graph $G = (V, A)$, sets $X = \{0\}$, $Y = \{0, 1\}$.
QUESTION: Are there functions $f : V \rightarrow X$, $g : A \rightarrow Y$, such that $\forall v \in V$: there is one unique edge $(u, v) \in A$ with $f((u, v)) = 1$ and there is one unique edge $(v, w) \in A$ with $f((v, w)) = 1$.

Q.E.D.

Theorem 5.16 PARTITION INTO HAMILTONIAN SUBGRAPHS $\in$ *ECC*.

Proof. Use the technique, outlined in section 5.2.3. **Q.E.D.**

5.2.13 Partition into forests [GT 14]

Theorem 5.17 PARTITION INTO FORESTS $\in$ *ECC*.

Proof. Use that the problem is equivalent 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.**

5.2.14 Partition into cliques [GT 15]

In [24] it is shown, that this problem is solvable in linear time for graphs with bounded treewidth, and bounded degree. This can also be shown with the following result.

Theorem 5.18 PARTITION INTO CLIQUES $\in$ 1-*LCC*.

Proof. We transform the problem to the following problem:

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

1. $\forall u \in V$: for all pairs of neighbors v, w of u: if $g((u, v)) = g((u, w)) = 1$, then $(v, w) \in E$.
2. the number of $v \in V$ with: for all adjacent w: $(g((v, w) = 0$ or $v \prec w)$ is at most K.

(A vertex v, with for all adjacent w: $(g((v, w) = 0$ or $v \prec w)$, is the first (with respect to the ordering $\prec$) vertex of a clique in $G' = (V, \{e \in E \mid f(e) = 1\})$. Hence, the number of cliques is at most K. The remainder of the proof is left to the reader.) **Q.E.D.**

We remark that one can find a linear algorithm for this problem on graphs with given tree-decomposition with bounded tree-width, without a restriction on the degree of the nodes. (Hint: use lemma 5.23, and make for each node $i \in I$ a table, with for each subset $S \subseteq X_i$ we store a number $c(S)$, which denotes the minimum number of cliques, in which the subgraph of G, induced by the set $\{v \in X_j \mid j \text{ is a descendant of } i,\ v \notin X_i\} \cup S\} \subseteq V\}$ can be partioned.)

5.2.15 Partition into perfect matchings [GT 16]

In [24] it is shown that the problem is solvable in linear time for graphs with a given treedecomposition with bounded treewidth, and a bounded degree, when we fix K, the maximum number of perfect matchings. For variable K, and no degree bound on the graphs, one obtains polynomial time algorithms with the following theorem.

Theorem 5.19 PARTITION INTO PERFECT MATCHINGS $\in$ *ECC.*

Proof. Transform the problem to the following problem:

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

1. $\forall v \in V$: $f_2(v)$ is a neighbor of v.
2. $\forall e = (v, w) \in E$: (if $f_2(v) = w$ then $f_2(w) = v$ and $f_1(v) = f_1(w)$) and (if $f_2(w) = v$ then $f_2(v) = w$ and $f_1(v) = f_1(w)$).
3. $\forall e = (v, w) \in E$: if $f_1(v) = f_1(w)$, then $f_2(v) = w$ and $f_2(w) = v$.

(It follows that for all $v \in V$ the vertex $w = f_2(v)$ is the unique neighbor with $f_1(v) = f_1(w)$. Hence, for all $i \leq K$, the subgraph induced by $V_i = \{v \in V \mid f_1(v) = i\}$, is a perfect matching.) **Q.E.D.**

5.2.16 Covering by cliques [GT 17]

Theorem 5.20 COVERING BY CLIQUES $\in$ *LCC.*

Proof. We use a technique, somewhat similar to that in section 5.2.14. Note that each vertex needs to be involved in at most $d = \text{degree}(G)$ cliques. We denote the set of subsets of a set S, with cardinality at most d by $\mathcal{P}^d(S)$. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \mathcal{P}^d(\{1, \ldots, K\})$, $Y = \{0\}$, with $K \leq |V|$, and a total ordering $\prec$ on V.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall u \in V$: for all pairs of neighbors v, w of u and for all $i, 1 \leq i \leq K$: if $i \in f(u)$ and $i \in f(v)$ and $i \in f(w)$, then $(v, w) \in E$.
2. $\forall e = (v, w) \in E$: $f(v) \cap f(w) \neq \emptyset$.
3. We require that: $\sum_{v \in V} val(v) \leq K$, where each vertex $v \in V$, val(v) denotes the number of $i, 1 \leq i \leq K$, such that there is no neighbor w of v, with $w \prec v$ and $i \in f(w)$.

The details are left to the reader. **Q.E.D.**

5.2.17 Covering by complete bipartite subgraphs [GT 18]

Similar (but slightly more complicated) to the proof of theorem 5.20 one shows the following result. (The main difference is that one has to look to vertices with distance at most 2, instead of only neighbors.)

Theorem 5.21 COVERING BY COMPLETE BIPARTITE SUBGRAPHS $\in$ *LCC.*

5.2.18 Clique [GT 19]

In [24] it was shown that CLIQUE is solvable in linear time, for graphs with bounded degree and given tree-decomposition with bounded treewidth. The following result shows that the bound on the degree can be avoided.

Theorem 5.22 CLIQUE $\in$ 2-*ECC.*

Proof. Transform the problem to the following problem:

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

1. $\sum_{v \in V} f(x) = K$.
2. the number of edges $e = (v, w) \in E$ with $f(v) = f(w) = 1$ is exactly $\frac{1}{2}K \cdot (K + 1)$.

Q.E.D.

However, we note that there is a very simple, and (for small values of k) efficient linear algorithm, to determine whether a graph with a given tree-decomposition with tree-width at most k contains a clique with l vertices. The algorithm is suggested by the following lemma. (Basically, one can look at each $i \in I$, and see whether the subgraph of G, induced by X_i contains a clique with l vertices.)

Lemma 5.23 *Let $(\{X_i \mid i \in I\}, T = (I, F))$, be a tree-decomposition of $G = (V, E)$. Suppose $W \subseteq V$ forms a clique in G. Then $\exists i \in I$: $W \subseteq X_i$.*

Proof. Use induction to the cliquesize $|W|$. For $|W| \leq 2$, the result follows directly. Suppose the lemma holds up to cliquesize $l - 1$. Consider a clique $W \subseteq V$, with $|W| = l$, and suppose the lemma does not hold for W. Choose a vertex $w \in W$, and let $W' = W - \{w\}$. Let $I' \subseteq I$ be the set $\{i \in I \mid W' \subseteq I\}$. By induction $I' \neq \emptyset$. Note that $w \in X_i \Rightarrow i \notin I'$. Now choose a node $i' \in I'$, and a node $i \in I$, with $w \in X_i$. Consider the path in T from i' to i. Let i'' be the last node on this path with $i'' \in I$, and let i''' be the next node on this path. Now, for every $w' \in W'$, there must be a vertex $j_{w'}$, with $\{w, w'\} \subseteq X_{j_{w'}}$. Consider the path from i'' to $j_{w'}$. We consider two cases.

Case 1: This path does not use i'''. In this case, the path in T from i to $j_{w'}$ uses i''. Now $w \in X_i$, $w \in X_{j_{w'}}$, hence $w \in X_{i''}$, contradiction.

Case 2: This path uses i'''. Now we have $w' \in X_{i''}$, $w' \in X_{j_{w'}}$, hence $w' \in X_{i'''}$.

It follows that for all $w' \in W'$: $w' \in X_{i'''}$, hence $i''' \in I'$, which contradicts the assumption that i'' was the last node on the path from i to i', that was in I'. **Q.E.D.**

5.2.19 Independent set [GT 20]

In [3] it is shown that this problem can be solved in linear time, for graphs with a given tree-decomposition with bounded treewidth. This result can also obtained with the following theorem.

Theorem 5.24 INDEPENDENT SET $\in$ 1-*ECC*.

Proof. Similar to the proof of theorem 5.2.1. **Q.E.D.**

5.2.20 Induced path [GT 23]

For the variant of this problem, where the minimum pathlength is fixed, Scheffler and Seese [24] proved solvability in linear time for graphs with bounded degree and given treedecomposition with bounded treewidth. One can also proof the following result.

Theorem 5.25 INDUCED PATH $\in$ *ECC*.

Proof. Transform to the following problem.

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

1. $\forall v \in V$: $f_2(v)$ is a neighbor of v, and $f_3(v)$ is a neighbor of v.
2. $\forall e = (v, w) \in V$: if $f_1(v) \neq 0$ and $f_1(w) \neq 0$, then $(|f_1(v) - f_1(w)| = 1$, and if $f_1(v) = f_1(w) + 1$, then $f_2(v) = w$ and $f_3(w) = v$ and if $f_1(w) = f_1(v) + 1$, then $f_2(w) = v$ and $f_3(v) = w)$.
3. $\forall (v, w) \in E$: if $1 \leq f_1(v) < K$ and $f_3(v) = w$ then $f_1(w) = f_1(v) + 1$.
4. $\forall (v, w) \in E$: if $1 < f_1(v) \leq K$ and $f_2(v) = w$ then $f_1(w) + 1 = f_1(v)$.
5. there is at least 1 vertex $v \in V$ with $f_1(v) \neq 0$.

(Here $f_1(v) = 0$, if v is not in V'; otherwise $f_1(v)$ denotes the number of v on the induced path: start with numbering one end of the path by 1, number the next vertex 2, etc. $f_2(v)$ denotes the vertex before v on the path; $f_3(v)$ the vertex after v.) The remaining details are left to the reader. **Q.E.D.**

One can modify this proof in order to obtain the following result:

Theorem 5.26 INDUCED PATH *with fixed pathlength* $K \in$ 1-*LCC.*

5.2.21 Balanced complete bipartite subgraph [GT 24]

With a technique, similar to that used for the CLIQUE problem, we show:

Theorem 5.27 BALANCED COMPLETE BIPARTITE SUBGRAPH $\in$ 3-*ECC.*

Proof. Transform to the following problem.

INSTANCE: Graph $G = (V, E)$, sets $X = \{0, 1, 2\} * V * V$, $Y = \{0\}$, positive integer $K \leq |V|$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. the number of vertices $v \in V$ with $f(v) = 1$ is exactly K.
2. the number of vertices $v \in V$ with $f(v) = 2$ is exactly K.
3. the number of edges $(u, v) \in E$ with $f(u) = 1$ and $f(v) = 2$ is exactly K^2.

Q.E.D.

A polynomial algorithm for this problem for graphs with a constant bound on the treewidth is easily obtained, by noting that a graph with treewidth $\leq k$ cannot have a balanced complete bipartite subgraph with $\geq 2k + 2$ vertices. Without much extra effort, one can obtain a linear algorithm for this problem, using that for graphs with treewidth $\leq k$, we may assume that K is bounded by a constant.

5.2.22 Bipartite subgraph [GT 25]

Theorem 5.28 BIPARTITE SUBGRAPH $\in$ 1-*ECC*.

Proof. Use the equivalence of the problem to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \{0, 1\}$, $Y = \{0\}$, positive integer $K \leq |V|$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that the number of edges $e = (v, w) \in E$ with $f(v) \neq f(w)$ is at least K.

The latter problem is easily seen to be a basic 1-ECC problem. **Q.E.D.**

5.2.23 Degree-bounded connected subgraph [GT 26]

Theorem 5.29 DEGREE-BOUNDED CONNECTED SUBGRAPH $\in$ *LCC*.

Proof. Use the equivalence of the problem to the following problem:

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

1. there is exactly one $v \in V$ with $f(v) = 0$.
2. for each $v \in V$, the number of adjacent edges e with $g(e) = 1$ is at most d.
3. for each vertex $v \in V$, with $f(v) \neq 0$, there is an adjacent vertex $w \in N_1(v)$, with $f(w) = f(v) - 1$, and $g((v, w)) = 1$.
4. $\sum_{e \in E} f(e) \geq K$.

(The graph $G' = (V, \{e \in E \mid f(e) = 1\})$ is connected, because each vertex $w \in V$ has a path (with length $f(w)$) in this graph to the unique vertex v with $f(v) = 0$.) **Q.E.D.**

5.2.24 Transitive subgraph [GT 29]

Theorem 5.30 TRANSITIVE SUBGRAPH $\in$ 1-*LCC*.

Proof. Transform to the following problem:

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

1. for all $v \in V$: for all u, with $(u, v) \in A$, and all w, with $(v, w) \in A$: if $g((u, v)) = g((v, w)) = 1$, then $(u, w) \in A$, and $g((u, w)) = 1$.
2. $\sum_{e \in E} g(e) \geq K$.

Q.E.D.

5.2.25 Cubic subgraph [GT 32]

In [24] this problem is shown to be solvable in linear time, for graphs with given tree-decomposition with bounded treewidth, and bounded degree. One can also show this result with the following theorem.

Theorem 5.31 CUBIC SUBGRAPH $\in$ 1-*LCC*.

Proof. Transform to the following problem:

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

1. for each vertex $v \in V$, there are either exactly 3, or exactly 0 adjacent edges e with $g(e) = 1$.
2. $\sum_{e \in E} g(e) \geq 1$.

Q.E.D.

Theorem 5.32 CUBIC SUBGRAPH $\in$ *ECC*.

Proof. Let $\mathcal{P}_3(V)$ denote the set of all subsets of V, that contain exactly 3 elements. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \mathcal{P}_3(V) \cup \{\emptyset\}$, $Y = \{0, 1\}$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: $w \in f(v) \Rightarrow w$ is a neighbor of v.
2. for all $e = (v, w) \in E$: if $w \in f(v)$ or $v \in f(w)$ then $g(e) = 1$.
3. for all $e = (v, w) \in E$: if $g(e) = 1$, then $w \in f(v)$ and $v \in f(w)$.
4. $\sum_{e \in E} g(e) \geq 1$.

Q.E.D.

5.2.26 Hamiltonian completion [GT 34]

Theorem 5.33 HAMILTONIAN COMPLETION $\in$ *ECC.*

Proof. We assume that $K \geq 1$. If $K = 0$, then see the HAMILTONIAN CIRCUIT problem.

Note that a graph $G = (V, E)$ has a Hamiltonian completion with $k \leq K$ extra edges, if and only if one can partition the vertices in V into $k \leq K$ disjoint subsets $V_1, \ldots, V_k$, with for all i, $1 \leq i \leq k$, V_i induces a subgraph of G which contains a Hamiltonian path. We now use a technique, similar to the technique used with e.g. INDUCED PATH.

Transform to the following problem:

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

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

5. $\forall v \in V$, $f_1(v) = 1$ or $f_2(v)$ is a neighbor of v.
6. the number of vertices v, with $f_1(v) = 1$ is at most K.

The remaining details of the proof are left to the reader. **Q.E.D.**

5.2.27 Hamiltonian circuit [GT 37] and variants

Hamiltonian circuit is solvable in linear time, for graphs with a given tree-decomposition with bounded treewidth (see [3]). Probably, this is an example, where the method of Arnborg and Proskurowski gives better results than the method of this paper, as we only were able to proof that HAMILTONIAN CIRCUIT $\in$ ECC (hence giving polynomial, instead of linear algorithms). The proof is very simple.

Theorem 5.34 HAMILTONIAN CIRCUIT $\in$ *ECC.*

Proof. Transform to the following problem:

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

1. $\forall v \in V$: $f_2(v)$ is a neighbor of v.
2. $\forall (v, w) \in V$: if $f_2(v) = w$, then $f_1(v) = (f_1(w)+1) \bmod |V|$.
3. $\forall (v, w) \in V$: if $f_2(w) = v$, then $f_1(w) = (f_1(v)+1) \bmod |V|$.

Q.E.D.

Of course, the related problems HAMILTONIAN PATH, DIRECTED HAMILTONIAN CIRCUIT and DIRECTED HAMILTONIAN PATH can be handled in the same way, with only minor variations. (For (DIRECTED) HAMILTONIAN PATH see e.g. HAMILTONIAN COMPLETION.) The proofs are left to the reader.

Theorem 5.35 *1.* HAMILTONIAN PATH $\in$ *ECC.*

2. DIRECTED HAMILTONIAN CIRCUIT $\in$ *ECC.*

3. DIRECTED HAMILTONIAN PATH $\in$ *ECC.*

5.2.28 Subgraph Isomorphism [GT 48]

We will consider some different subproblems of this problem. First note that the following problem is NP-complete (by transformation from 3-PARTITION).

INSTANCE: Tree $G = (V, E)$ with degree$(G) \leq 3$, forest $H = (W, F)$, with degree$(H) \leq 2$.
QUESTION: Does G contain a subgraph isomorphic to H?

Thus, we cannot expect to obtain polynomial algorithms for SUBGRAPH ISOMORPHISM for graphs with bounded tree-width, unless we assume that H is connected (or unless P=NP). One also must require that G has a bounded tree-width. (If we only have that H has a bounded tree-width, then again we have an NP-complete variant of the problem; e.g. use a transformation from HAMILTONIAN PATH.)

So, let us now consider the case, where H is connected, and G has a bounded tree-width, i.e. G is the graph, appearing in the instance of the graph decision problem, and H is "hidden" somewhere in the other information I. This version of the problem is in LCC. This can be seen by transformation to the following problem:

INSTANCE: Graph $G = (V_G, E_G)$, connected graph $H = (V_H, E_H)$, sets $X = V_H \cup \{0\}$, $Y = \{0\}$, vertex $w \in V_H$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V_G$: if $f(v) \neq 0$, then for every $w' \in V_H$, that is adjacent to $f(v)$, there is a $v' \in V_G$, adjacent to v, with $f(v') = w$.
2. there is exactly one vertex $v \in V_G$, with $f(v) = w$.

It follows that the problem to determine whether a given connected graph H is isomorphic to a given graph G can be solved in polynomial time, for graphs G with constant bounded treewidth and constant bounded degree. For graphs with no bounds on the degree, such a result would imply that P=NP, because SUBGRAPH ISOMORPHISM is NP-complete, for G and H connected, outerplanar graphs [27]. (Recall that each outerplanar graph has treewidth at most 2.) (For a discussion of similar results, see section 6.2.)

As a curiosity we mention, that the fact, that the problem to decide whether a graph G has bandwidth at most k, for some constant k, is solvable in polynomial time, follows as a corollary. (This result was first obtained by Saxe [23].) Let $G_{k,n}$ be the maximal graph on n vertices with bandwidth k, i.e. $G_{k,n} = (V_n, E_{k,n})$, with $V_n = \{1, 2, \ldots, n\}$ and $E_{k,n} = \{(i,j) \mid i,j \in V_n \wedge |i-j| \leq k\}$. The following observations was already made by Saxe [23].

Lemma 5.36 (Saxe [23]) *Let $G = (V,E)$, with $|V| = n$. Then bandwidth(G) $\leq k$, if and only if G is isomorphic to a subgraph of $G_{k,n}$.*

The treewidth of $G_{k,n}$ also is at most k. The degree of $G_{k,n}$ is at most $2k$. Hence, it follows, that one can decide in polynomial time, for each connected graph G (and hence, also for each G, that is not connected, by applying the algorithm to each connected component of G), whether G is isomorphic to a subgraph of $G_{k,n}$, and thus, by lemma 5.36, whether bandwidth$(G) \leq k$.

5.2.29 Graph contractability [GT 51]

We only consider this problem for fixed graphs H. With induction to the number of edge-contractions one can proof the following lemma.

Lemma 5.37 *Let $G = (V_G, E_G)$, $H = (V_H, E_H)$ be graphs. We can obtain a graph isomorphic to H from G by a sequence of edge contractions, if and only if we can associate with each vertex $v \in V_H$ a set of vertices $F(v) \subseteq V_G$, such that*

1. *For all $v \in V$: the subgraph of G, induced by $F(v)$ is connected.*
2. *$v \neq w \Leftrightarrow F(v) \cap F(w) = \emptyset$.*
3. *$\bigcup_{v \in V_H} F(v) = V_G$.*
4. *For all $v, w \in V_H$: $(v,w) \in E_H \Leftrightarrow (\exists v' \in F(v), w' \in F(w) : (v', w') \in E_G)$.*

We leave the proof to the reader.

Theorem 5.38 *For all graphs H,* GRAPH CONTRACTABILITY *to $H \in LCC$.*

Proof. Transform to the following problem:

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

1. For each $v \in V_H$, we have a condition: There is exactly one $w \in V_G : f(w) = (v, 0)$.
2. For each $(v_1, v_2) \in E_H$, we have a condition: $\exists (w_1, w_2) \in E_G : f_1(w_1) = v_1 \wedge f_1(w_2) = v_2$.
3. $\forall (v, w) \in E_G : f_1(v) = f_1(w)$ or $(f_1(v), f_1(w)) \in E_H$.
4. $\forall v \in V_G : f_2(v) = 0$ or v has a neighbor w with $f_1(v) = f_1(w)$ and $f_2(w) < f_2(v)$.

(f represents F^{-1}.) Note that we have $|V_H| + |E_H| + 2$ conditions! (This is constant, because H is fixed.) **Q.E.D.**

With the technique, outlined in section 5.2.3, one obtains also the following result:

Theorem 5.39 *For all graphs H,* GRAPH CONTRACTABILITY *to $H \in ECC$.*

5.2.30 Graph homomorphism [GT 52]

This problem, for fixed graphs H, can be dealt with, similar to GRAPH CONTRACTABILITY.

Lemma 5.40 *Let $G = (V_G, E_G)$, $H = (V_H, E_H)$ be graphs. We can obtain a graph isomorphic to H from G by a sequence of identifications of non-adjacent vertices, if and only if we can associate with each vertex $v \in V_H$ a set of vertices $F(v) \subseteq V_G$, such that*

1. *For all $v \in V : F(v)$ is an independent set in G.*
2. *$v \neq w \Leftrightarrow F(v) \cap F(w) = \emptyset$.*
3. *$\bigcup_{v \in V_H} F(v) = V_G$.*
4. *For all $v, w \in V_H$: $(v, w) \in E_H \Leftrightarrow (\exists v' \in F(v), w' \in F(w) : (v', w') \in E_G)$.*

Again, we leave the proof to the reader.

Theorem 5.41 *For all graphs H,* GRAPH HOMOMORPHISM *to $H \in$ LCC.*

Proof. Transform to the following problem:

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

1. For each $v \in V_H$, we have a condition: There is exactly one $w \in V_G : f(w) = v$.
2. For each $(v_1, v_2) \in E_H$, we have a condition: $\exists (w_1, w_2) \in E_G : f(w_1) = v_1 \wedge f(w_2) = v_2$.
3. $\forall (v, w) \in E_G : (f(v), f(w)) \in E_H$.

Q.E.D.

Theorem 5.42 *For all graphs H,* GRAPH CONTRACTABILITY *to $H \in$ ECC.*

Proof. Again, use the technique outlined in section 5.2.3. **Q.E.D.**

5.2.31 Graph grundy numbering [GT 56]

Theorem 5.43 GRAPH GRUNDY NUMBERING $\in$ 1-*LCC.*

Proof. Note that the function f never has to take values beyond the range $\{1, \ldots, d+1\}$, where $d = \text{degree}(G)$. Now one can proof membership in 1-LCC by standard methods. **Q.E.D.**

5.2.32 Kernel [GT 57]

Theorem 5.44 KERNEL $\in$ 1-*LCC.*

Proof. Transform to the following problem:

INSTANCE: Directed graph $G = (V, A)$, sets $X = \{0, 1\}$, $Y = \{0\}$.
QUESTION: Are there functions $f : V \to X$, $g : A \to Y$, such that

1. $\forall e = (v, w) \in A$: $f(v) = 0$ or $f(w) = 0$.

2. $\forall v \in V$: if $f(v) = 0$, then there is an edge $(u, v) \in A$, with $f(u) = 1$.

Q.E.D.

Theorem 5.45 KERNEL $\in$ *ECC.*

Proof. Transform to the following problem:

INSTANCE: Directed graph $G = (V, A)$, sets $X = \{0, 1\} * V$, $Y = \{0\}$.
QUESTION: Are there functions $f : V \to X$, $g : A \to Y$, such that

1. $\forall e = (v, w) \in A$: $f_1(v) = 0$ or $f_1(w) = 0$.
2. $\forall v \in V$: $f_1(v) = 1$ or $(f_2(v), v) \in A$.
3. $\forall e = (v, w) \in A$: $f_1(w) = 1$ or $f_2(w) \neq v$ or $f_1(v) = 1$.

Q.E.D.

5.2.33 K-closure [GT 58]

Theorem 5.46 K-CLOSURE $\in$ 1-*ECC.*

Proof. Immediate. **Q.E.D.**

5.2.34 Intersection graph basis [GT 59]

This problem is equivalent to COVERING BY CLIQUES, and hence in LCC.

5.2.35 Degree constrained spanning tree [ND 1]

Scheffler and Seese [24] prove that this problem is solvable in linear time for graphs with bounded degree and given tree-decomposition with bounded treewidth. A similar, but weaker result can be obtained with the following result.

Theorem 5.47 DEGREE CONSTRAINED SPANNING TREE $\in$ *LCC.*

We omit the proof of this theorem. If we fix the maximum degree of the spanning tree K, then the problem is in ECC, (and hence can be solved in polynomial time for arbitrary graphs with constant bounded tree-width, without a restriction on the degree of G.) We will call the subproblem of DEGREE CONSTRAINED SPANNING TREE where the maximum degree of the spanning tree must be K or less, "DEGREE K SPANNING TREE".

Theorem 5.48 DEGREE K SPANNING TREE $\in ECC$.

Proof. If we have a spanning tree $T = (V, F)$ of $G = (V, E)$, with degree$(T) \leq K$, then we can chose some arbitrary vertex $v^* \in V$ as root of the tree, and then we can associate with each vertex, a number, denoting its distance to the root, a vertex denoting its father in the tree, (except for the root), and a subset of V with at most $K - 1$ vertices in it (except for the root, where it may contain K vertices), denoting the sons of the vertex in the tree. (By chosing v^* some vertex with degree less than K, we may assume each vertex has at most $K - 1$ sons.)

Thus, we can transform the problem to the following problem:

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

1. There is exactly one vertex with $f_1(v) = 0$.
2. $\forall v \in V : f_2(v) \in N_1(v) - \{v\}$ or $f_1(v) = 0$.
3. $\forall v \in V : f_3(v) \subseteq N_1(v) - \{v\}$.
4. $\forall (v, w) \in E$: if $f_2(w) = v$, then $f_1(w) = f_1(v) + 1$ and $w \in f_3(v)$.
5. $\forall (v, w) \in E$: if $f_2(v) = w$, then $f_1(v) = f_1(w) + 1$ and $v \in f_3(w)$.

We leave the details for the reader. **Q.E.D.**

5.2.36 Maximum leaf spanning tree [ND 2]

Scheffler and Seese [24] showed linear time solvability for the subproblem where the minimum number of leaves is some given constant, for graphs with given constant-width tree-decompositions, and constant bounded degree.

We do not obtain linear time algorithm, but polynomial time algorithms. However, we do not need a bound on the degree of the graph, and the minimum number of leaves may be variable.

Theorem 5.49 MAXIMUM LEAF SPANNING TREE $\in$ *ECC.*

Proof. Again, we associate with each vertex a number, that denotes its distance to the root, and another vertex, that denotes its father in the tree. $f_3(v) = 1$, if and only if v is a leaf in the tree, i.e. if and only if there is no other node w, with v the father of W in the tree. We transform the problem to the following problem:

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

1. There is exactly one vertex with $f_1(v) = 0$.
2. $\forall v \in V : f_2(v) \in N_1(v) - \{v\}$ or $f_1(v) = 0$.
3. $\forall (v, w) \in E$: if $f_2(w) = v$, then $f_1(w) = f_1(v) + 1$ and $f_3(v) = 0$.
4. $\forall (v, w) \in E$: if $f_2(v) = w$, then $f_1(v) = f_1(w) + 1$ and $f_3(w) = 0$.
5. $\sum_{v \in V} f_3(v) \geq K$.

Again, we leave the details for the reader. **Q.E.D.**

5.2.37 Shortest total path length spanning tree [ND 3]

Theorem 5.50 SHORTEST TOTAL PATH LENGTH SPANNING TREE $\in$ *LCC.*

Proof. Note that the sum over all pairs of vertices $u, v \in V$, of the length of the path from u to v in a spanning tree $T = (V, F)$ equals the sum over all edges (u, v) in T of the number of paths from vertices $w \in V$ to $x \in V$, that use this edge. The latter number equals the product of the number of vertices in the two subtrees of T, obtained by removing the edge (u, v) from T. So we map each edge $e = (u, v)$ on a pair of two numbers $g_e(u), g_e(v)$, denoting the number of vertices in the subtree with u, and v respectively as

root. We count the sum of $g_e(u) \cdot g_e(v)$ over all edges $e = (u, v)$, that are chosen to be in the spanning tree T. The other techniques are similar as in section 5.2.36. **Q.E.D.**

5.2.38 Bounded diameter spanning tree [ND 4]

Theorem 5.51 BOUNDED DIAMETER SPANNING TREE $\in$ *ECC.*

Proof. We associate each vertex v with a number $(f_3(v))$, that denotes the maximum distance of the vertex to a leaf, in its subtree. The next vertex on the path to this leaf is denoted with $f_4(v)$. The requirement that the diameter of the graph is bounded by D now can be expressed with the conditions 9 and 10 below. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \{0, 1, \ldots, |V|-1\} * V * \{0, 1, \ldots, |V| - 1\} * V$, $Y = \{0, 1\}$, positive integer $D \leq |V|$, weight $w(e) \in N^+$ for each $e \in E$, positive integer B.

QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. There is exactly one vertex with $f_1(v) = 0$.
2. $\forall v \in V : f_2(v) \in N_1(v) - \{v\}$ or $f_1(v) = 0$.
3. $\forall v \in V : f_4(v) \in N_1(v) - \{v\}$ or $f_3(v) = 0$.
4. $\forall e = (v, w) \in E$: if $f_2(w) = v$, then $f_1(w) = f_1(v) + 1$ and $g(e) = 1$ and $f_3(v) \geq f_3(w) + 1$.
5. $\forall e = (v, w) \in E$: if $f_2(v) = w$, then $f_1(v) = f_1(w) + 1$ and $g(e) = 1$ and $f_3(w) \geq f_3(v) + 1$.
6. $\forall e = (v, w) \in E$: if $f_4(w) = v$, then $f_3(w) = f_3(v) + 1$ and $g(e) = 1$.
7. $\forall e = (v, w) \in E$: if $f_4(v) = w$, then $f_3(v) = f_3(w) + 1$ and $g(e) = 1$.
8. $\forall e = (v, w) \in E$: if $g(e) = 1$, then $|f(v) - f(w)| \leq 1$.
9. $\forall e = (v, w) \in E$: if $g(e) = 1$ and not $(f_4(v) = w$ or $f_4(w) = v)$, then $f_3(v) + f_3(w) + 1 \leq D$.
10. $\forall v \in V$: $f_3(v) \leq D$.
11. $\sum_{e \in E} g(e) \cdot w(e) \leq B$.

Q.E.D.

5.2.39 Isomorphic spanning tree [ND 8]

This problem is a special case of subgraph isomorphism. From the discussing in section 5.2.28 the following result follows.

Theorem 5.52 ISOMORPHIC SPANNING TREE $\in$ *LCC.*

5.2.40 Bounded component spanning forest [ND 10]

We consider the version of the problem where the weight $w(v) \in N^+$ of the vertices are given in unary notation, i.e. $w(v) \leq s(D)$. (This problem is also NP-complete.) We have the following result.

Theorem 5.53 BOUNDED COMPONENT SPANNING FOREST *with weights in unary notation* $\in$ *LCC.*

Proof. In each component one can choose a spanning tree, and a root in this tree. We associate with every vertex v its father in the tree $f_1(v)$ (except for the root), and the sum of the weights of all vertices in its subtree $f_2(v)$. Here, we now do not need to use the distance of a vertex to the root. A vertex v, with $f_1(v) \notin N_c(v) - \{v\}$, is assumed to be a root of a spanning tree. (We use that $f_2(v) > f_2(w)$, for all sons v of w.)

Transform the problem to the following problem:

INSTANCE: Graph $G = (V, E)$, weight $w(v)$ for every $v \in V$, sets $X = V * \{0, 1, \ldots, \sum_{v \in V} w(v)\}$, $Y = \{0\}$, positive integer $K \leq |V|$, positive integer B.

QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: $f_2(v) = \sum_{w \in N_1(v) - \{v\}; f_1(w) = v} (f_2(w) + w(w))$.
2. $\forall v \in V$: $f_2(v) + w(v) \leq B$.
3. The number of $v \in V$, with $f_1(v) \notin N_1(v) - \{v\}$ is at most K.

Q.E.D.

With a more precise argument, one can show that the version of the problem, where B is any *fixed* positive integer, is in 1-ECC. (For instance, one can use that each vertex has at most $B - 1$ sons in a spanning tree.)

5.2.41 Steiner tree in graphs [ND 12]

Scheffler and Seese [24] proved that the problem of determining whether a given graph G has a Steiner tree of weight B or less for fixed B and a fixed number k of given vertices, is solvable in linear time for graphs with bounded degree and given tree-decomposition with bounded treewidth. With our techniques, we have polynomial time algorithms (instead of linear), but do not restrict the degree of the graphs, and have variable B and k.

Theorem 5.54 STEINER TREE IN GRAPHS $\in$ *ECC.*

Proof. This can be shown with techniques, quite similar to those used in previous sections on spanning-tree problems. **Q.E.D.**

5.2.42 Graph partitioning [ND 14]

Theorem 5.55 GRAPH PARTITIONING *with weights given in unary notation* $\in$ *LCC.*

Proof. First we note, that without altering the problem, we can require that each set V_i induces a connected subgraph of $G = (V, E)$. We handle this problem more or less similar to BOUNDED COMPONENT SPANNING FOREST. Components are 'named' by the vertex that is the root. Each vertex has this 'name' associated with it in $f_3(v)$.

Transform the problem to the following problem:

INSTANCE: Graph $G = (V, E)$, weights $w(v)$ for every $v \in V$, $l(e)$ for every $e \in E$, sets $X = V * \{0, 1, \ldots, \sum_{v \in V} w(v)\} * V$, $Y = \{0\}$, positive integer $K \leq |V|$, positive integer B.

QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: $f_2(v) = \sum_{w \in N_1(v) - \{v\}; f_1(w) = v} (f_2(w) + w(w))$.
2. $\forall v \in V$: $f_2(v) + w(v) \leq B$.
3. $\forall v \in V$: $f_3(v) = v$, or ($f_1(v) \in N_1(v) - \{v\}$ and $f_3(v) = f_3(f_1(v))$).
4. $$\sum_{(v,w) \in E; f_3(v) \neq f_3(w)} l((v, w)) \leq J.$$

Q.E.D.

5.2.43 Acyclic partitioning [ND 15]

This problem can be handled more or less similar as GRAPH PARTITIONING.

Theorem 5.56 ACYCLIC PARTITIONING *with weights given in unary notation* $\in$ *LCC.*

Proof. In order to represent the acyclic property of the graph G', we can associate with each component a number (with f_4); and require for each edge $(v,w) \in E$, that $f_4(v) \leq f_4(w)$. (Note that $N_1(v)$ denotes the vertices adjacent ot equal to v, without regard to the direction of the edges.)

Transform the problem to the following problem:

INSTANCE: Directed graph $G = (V,A)$, weights $w(v)$ for every $v \in V$, $l(e)$ for every $e \in A$, sets $X = V * \{0,1,\ldots,\sum_{v\in V} w(v)\} * V$, $Y = \{0\}$, positive integer $K \leq |V|$, positive integer B.

QUESTION: Are there functions $f : V \to X$, $g : A \to Y$, such that

1. $\forall v \in V$: $f_2(v) = \sum_{w\in N_1(v)-\{v\};f_1(w)=v}(f_2(w) + w(w))$.
2. $\forall v \in V$: $f_2(v) + w(v) \leq B$.
3. $\forall v \in V$: $f_3(v) = v$, or $(f_1(v) \in N_1(v) - \{v\}$ and $f_3(v) = f_3(f_1(v)))$.
4. $\forall (v,w) \in E$: $f_3(v) = f_3(w) \Leftrightarrow f_4(v) = f_4(w)$.
5. $\forall (v,w) \in E$: $f_4(v) \leq f_4(w)$.
6. $$\sum_{(v,w)\in A;f_3(v)\neq f_3(w)} l((v,w)) \leq J.$$

Q.E.D.

5.2.44 Max cut [ND 16]

Theorem 5.57 MAX CUT $\in$ 1-*ECC.*

Proof. Transform to the following problem:

INSTANCE: Graph $G = (V,E)$, sets $X = \{0,1\}$, $Y = \{0\}$, positive integer K, a weight $w(e)$ for each $e \in E$.

QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

$$\sum_{e=(v,w)\in E} |f(v) - f(w)| \cdot w(e) \geq K$$

Q.E.D.

5.2.45 Minimum cut into bounded sets [ND 17]

Theorem 5.58 MINIMUM CUT INTO BOUNDED SETS $\in$ 2-*ECC*.

Proof. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = \{0, 1\}$, $Y = \{0\}$, positive integer K, a weight $w(e)$ for each $e \in E$, positive integer $B \leq |V|$, specified vertices $s, t \in V$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $f(s) = 0$.
2. $f(t) = 1$.
3. $|V| - B \leq \sum_{v\in V} f(v) \leq B$.
4. $\sum_{e=(v,w)\in E} |f(v) - f(w)| \cdot w(e) \geq K$.

Q.E.D.

5.2.46 Longest Circuit [ND 28]

Theorem 5.59 LONGEST CIRCUIT $\in$ *ECC*.

Proof. Transform to the following problem:

INSTANCE: Graph $G = (V, E)$, sets $X = (\{0, 1, \ldots, |V| - 1\} * V * V) \cup \{\epsilon\}$, $Y = \{0\}$, positive integer K, a length $l(e)$ for each $e \in E$.
QUESTION: Are there functions $f : V \to X$, $g : E \to Y$, such that

1. $\forall v \in V$: if $f(v) \neq \epsilon$, then $f_2(v)$ and $f_3(v)$ are neighbors of v.

2. $\forall(v,w),(w,v) \in V$: if $f(v) \neq \epsilon$ and $f_2(v) = w$, then $f(w) \neq \epsilon$ and $f_3(w) = v$ and ($f_1(v) = 0$ or $f_1(w) = f_1(v) - 1$).
3. $\forall(v,w),(w,v) \in V$: if $f(v) \neq \epsilon$ and $f_3(v) = w$, then $f(w) \neq \epsilon$ and $f_2(w) = v$ and ($f_1(w) = 0$ or $f_1(w) = f_1(v) + 1$).
4. There is exactly one vertex $v \in V$, with $f_1(v) = 0$.
5. $\sum l(e) \geq K$, where the sum is taken over all edges (v,w), with $f_2(v) = w$ and $f_3(w) = v$.

Q.E.D.

5.2.47 Longest Path [ND 29]

Theorem 5.60 LONGEST PATH $\in$ *ECC.*

Proof. Similar to the proof of theorem 5.59 **Q.E.D.**

5.2.48 Chromatic Index

Scheffler and Seese [24] show that this problem is solvable in linear time for graphs with given tree-decomposition with treewidth, bounded by a constant, and degree bounded by a constant. This result can also be obtained with the following theorem.

Theorem 5.61 CHROMATIC INDEX $\in$ 1-*LCC.*

Proof. By using that the chromatic index of a graph is either its maximum degree, or its maximum degree +1, hence is bounded by a constant, for constant degree graphs, it follows easily that CHROMATIC INDEX $\in$ 1-LCC. **Q.E.D.**

The CHROMATIC INDEX problem is solvable in polynomial time, (with a similar, but slightly more involved technique) for graphs with bounded tree-width, but without a bound on the degree [7].

VERTEX COVER	1 [3]	1 [3]
DOMINATING SET	1 [3]	1 [3]
DOMATIC NUMBER	1 [24]	'P'
CHROMATIC NUMBER	1 [3]	1 [3]
MONOCHROMATIC TRIANGLE	1 [24]	'1'
FEEDBACK VERTEX SET	P	P
FEEDBACK ARC SET	P	P
PARTIAL FEEDBACK ARC SET	1 (fixd L)	?
MINIMUM MAXIMAL MATCHING	1	P
PARTITION INTO TRIANGLES	1	P
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' [24]; 2	2; '1'
INDEPENDENT SET	1 [3]	1 [3]
INDUCED PATH	1 [24](fixd K); P	P
BALANCED COMPLETE BIPARTITE SUBGRAPH	3; '1'	3; '1'
BIPARTITE SUBGRAPH	1	1
DEGREE-BOUNDED CONNECTED SUBGRAPH	P	?
TRANSITIVE SUBGRAPH	1	?
CUBIC SUBGRAPH	1 [24]	P
HAMILTONIAN COMPLETION	P	P
HAMILTONIAN CIRCUIT	'1' [3]	'1' [3]
SUBGRAPH ISOMORPHISM for connected graphs	P (^{a}st)	N [27]
GRAPH CONTRACTABILITY to a fixed graph H	P	P
GRAPH HOMOMORPHISM to a fixed graph H	P	P

GRAPH GRUNDY NUMBERING	1	?
KERNEL	1	P
K-CLOSURE	1	1
INTERSECTION GRAPH BASIS	P	?
DEGREE K SPANNING TREE	'1' [24] (fixd k); P	P
MAXIMUM LEAF SPANNING TREE	'1' [24]	P
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, fixd B)
STEINER TREE IN GRAPHS	'1' [24] (fixd B,k); P	P
GRAPH PARTITIONING (weak version)	P	?
ACYCLIC PARTITIONING (weak version)	P	?
MAX CUT	1	1
MINIMUM CUT INTO BOUNDED SETS	2	2
LONGEST CIRCUIT	P	P
LONGEST PATH	P	P
CHROMATIC INDEX	1 [24]	'P' [7]

Table 1:
Overview of complexity results of several problems, restricted to graphs with given tree-decomposition with bounded treewidth. 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. Also solvable in polynomial time, for graphs with given tree-decomposition with logarithmic tree-width (and, in the case of column 1, degree bounded by a constant). '1': problem solvable in linear time. P: problem in LCC or ECC, hence solvable in polynomial time. 'P': Problem solvable in polynomial time. N: Problem NP-complete. ?: Open, whether problem solvable in polynomial time or not. (*): For Subgraph Isomorphism, the larger graph G must have the bounded tree-width. Other restrictions are given in the table.

6 Overview of results and final remarks

6.1 Overview of results

In table 1 we give an overview of the known results for the considered problems. For many other (NP-complete) graph decision problems, similar results can be obtained.

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 tree-width (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) [20]. One can show that ISOMORPHIC SPANNING TREE is NP-complete, when restricted to graphs with tree-width ≤ 3, by transformation from 3-PARTITION. From theorem 3.5 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 [27]. (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. (This result follows also from the result for ISOMORPHIC SPANNING TREE.)

For the OPTIMAL LINEAR ARRANGEMENT problem, Sudborough [25] 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).

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

Theorem 6.2 *LCC* $\subseteq$ *NP.*

Proof. One can guess f and g non-deterministically in polynomial time, and then check in polynomial time whether $\bigoplus_{v \in V}^{p} val_p(D, v, f|_{N_c(v)}, g|_{M_c(v)})$ $\in R_p$ or $\preceq K$, for $1 \leq p \leq m+1$. **Q.E.D.**

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.

The algorithms in this paper can be modified to run on a parallel machine, e.g. a EREW PRAM. One can find a tree-decomposition with treewidth at most a constant k in poly-logarithmic parallel time, with a polynomial number of processors. The dynamic programming algorithms of section 3 can be transformed to parallel algorithms, that use a polynomial number of processors (linear, for problems in 1-LCC, 1-ECC), and $O(\log n)$ time or $O(\log^2 n)$ time on a EREW PRAM. These results will be reported elsewhere.

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 and A. Proskurowski. *Linear Time Algorithms for NP-Hard Problems on Graphs Embedded in k-Trees.* TRITA-NA-8404, Department Of Numerical Analysis And Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1984.

[4] 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.

[5] 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.

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

[7] H. L. Bodlaender. A polynomial algorithm for Chromatic Index on graphs with bounded treewidth. Unpublished result.

[8] 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.

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

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

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

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] B. Monien and I. 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.

[20] 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*, Springer Verlag Lecture Notes in Comp. Science, Vol 226, 1986.

[21] 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.

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

[23] J. B. Saxe. Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Alg. Disc. Meth.*, 1:363–369, 1980.

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

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

[26] 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.

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

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

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

[30] 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.

Contents

1 Introduction **2**

2 Definitions and preliminary results **3**
2.1 Graph definitions 3
2.2 Algebraic definitions 5
2.3 Other notations 7
2.4 Graph decision problems 7
2.5 Local condition composition problems and edge condition composition problems 8

3 Polynomial time algorithms for LCC-problems and ECC-problems on graphs with bounded treewidth **13**

4 Small-degree polynomial time algorithms for subclasses of LCC and ECC **22**

5 Problems in LCC and ECC **24**
5.1 Some basic techniques 24
5.2 A list of problems in LCC and ECC 26
5.2.1 Vertex cover [GT 1] 26
5.2.2 Dominating set [GT 2] 27
5.2.3 Domatic Number [GT 3] 27
5.2.4 Chromatic Number [GT 4] 29
5.2.5 Monochromatic triangle [GT 6] 29
5.2.6 Feedback vertex set [GT 7] 30
5.2.7 Feedback arc set [GT 8] 30
5.2.8 Partial feedback edge set [GT 9] 31
5.2.9 Minimum maximal matching [GT 10] 32
5.2.10 Partition into triangles [GT 11] 33
5.2.11 Partition into Isomorphic subgraphs [GT 12] 33
5.2.12 Partition into Hamiltonian Subgraphs [GT 13] 34
5.2.13 Partition into forests [GT 14] 34
5.2.14 Partition into cliques [GT 15] 35
5.2.15 Partition into perfect matchings [GT 16] 36
5.2.16 Covering by cliques [GT 17] 36
5.2.17 Covering by complete bipartite subgraphs [GT 18] . . 37
5.2.18 Clique [GT 19] 37
5.2.19 Independent set [GT 20] 39

5.2.20 Induced path [GT 23] 39
5.2.21 Balanced complete bipartite subgraph [GT 24] 40
5.2.22 Bipartite subgraph [GT 25] 41
5.2.23 Degree-bounded connected subgraph [GT 26] 41
5.2.24 Transitive subgraph [GT 29] 42
5.2.25 Cubic subgraph [GT 32] 42
5.2.26 Hamiltonian completion [GT 34] 43
5.2.27 Hamiltonian circuit [GT 37] and variants 44
5.2.28 Subgraph Isomorphism [GT 48] 45
5.2.29 Graph contractability [GT 51] 46
5.2.30 Graph homomorphism [GT 52] 47
5.2.31 Graph grundy numbering [GT 56] 48
5.2.32 Kernel [GT 57] 48
5.2.33 K-closure [GT 58] 49
5.2.34 Intersection graph basis [GT 59] 49
5.2.35 Degree constrained spanning tree [ND 1] 49
5.2.36 Maximum leaf spanning tree [ND 2] 50
5.2.37 Shortest total path length spanning tree [ND 3] 51
5.2.38 Bounded diameter spanning tree [ND 4] 52
5.2.39 Isomorphic spanning tree [ND 8] 53
5.2.40 Bounded component spanning forest [ND 10] 53
5.2.41 Steiner tree in graphs [ND 12] 54
5.2.42 Graph partitioning [ND 14] 54
5.2.43 Acyclic partitioning [ND 15] 55
5.2.44 Max cut [ND 16] 55
5.2.45 Minimum cut into bounded sets [ND 17] 56
5.2.46 Longest Circuit [ND 28] 56
5.2.47 Longest Path [ND 29] 57
5.2.48 Chromatic Index 57

6 Overview of results and final remarks **60**
6.1 Overview of results 60
6.2 Problems, that are not in LCC, (unless P = NP) 60
6.3 Final remarks 61

CS-TR Scanning Project
Document Control Form

Date : 3/23/95

Report # LCS-TR-394

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 68(74-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☐ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MOB (1-2) UNNUMBERED TITLE PAGES	
(3-68) PAGES #'ED 1-66	
(69-71) SCANCONTRL, COVER, PRINTERS NOTES	
(72-74) TRGTS	

Scanning Agent Signoff:
Date Received: 3/23/95 Date Scanned: 3/30/95 Date Returned: 3/30/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 3/30/1995

M.I.T. Libraries
Document Services