Minimizing NLC-Width is NP-Complete
(Extended Abstract)

Frank Gurski and Egon Wanke

Heinrich-Heine-University Düsseldorf,
Institute of Computer Science, D-40225 Düsseldorf, Germany
{gurski-wg, wanke-wg}@acs.uni-duesseldorf.de

Abstract. We show that a graph has tree-width at most $4k-1$ if its line graph has NLC-width or clique-width at most k, and that an incidence graph has tree-width at most k if its line graph has NLC-width or clique-width at most k. In [9] it is shown that a line graph has NLC-width at most $k+2$ and clique-width at most $2k+2$ if the root graph has tree-width k. Using these bounds we show by a reduction from tree-width minimization that NLC-width minimization is NP-complete.

1 Introduction

The clique-width of a graph is defined by a composition mechanism for vertex-labeled graphs [7]. The operations are the vertex disjoint union, the addition of edges between vertices controlled by a label pair, and the relabeling of vertices. The clique-width of a graph G is the minimum number of labels needed to define it. The NLC-width of a graph is defined by a composition mechanism similar to that for clique-width [19]. Every graph of clique-width at most k has NLC-width at most k and every graph of NLC-width at most k has clique-width at most $2k$ [12]. The only essential difference between the composition mechanisms of clique-width bounded graphs and NLC-width bounded graphs is the addition of edges. In an NLC-width composition the addition of edges is combined with the union operation. This union operation applied to two graphs G and J is controlled by a set S of label pairs such that for every pair $(a,b) \in S$ all vertices of G labeled by a will be connected with all vertices of J labeled by b. Both concepts are useful, because it is sometimes much more comfortable to use NLC-width expressions instead of clique-width expressions and vice versa, respectively.

Clique-width and NLC-width bounded graphs are particularly interesting from an algorithmic point of view. A lot of NP-complete graph problems can be solved in polynomial time for graphs of bounded clique-width. For example, all graph properties expressible in monadic second order logic with quantifications over vertices and vertex sets (MSO_1-logic) are decidable in linear time on clique-width bounded graphs [6] if a corresponding decomposition for the graph is given as an input. The MSO_1-logic has been extended by counting mechanisms which allow the expressibility of optimization problems concerning maximal or minimal vertex sets [6]. All graph problems expressible in extended MSO_1-logic can be

D. Kratsch (Ed.): WG 2005, LNCS 3787, pp. 69–80, 2005.

solved in polynomial time on clique-width bounded graphs. Furthermore, there are a lot of NP-complete graph problems which are not expressible in extended MSO_1-logic (like Hamiltonicity, various partition problems, and bounded degree subgraph problems) but which can also be solved in polynomial time on clique-width bounded graphs [19,8,14,9].

The recognition problem for graphs of clique-width or NLC-width at most k for fixed integers k is still open for $k \geq 4$ and $k \geq 3$, respectively. Clique-width of at most 3 is decidable in polynomial time [4]. NLC-width of at most 2 is decidable in polynomial time [13]. Clique-width of at most 2 and NLC-width 1 is decidable in linear time [5]. In this paper we show that NLC-width minimization is NP-complete, which was open up to now.

The paper is organized as follows. In Section 2, we recall the definitions of clique-width, NLC-width, and tree-width. In Section 3, we show that a graph has tree-width at most $4k-1$ if its line graph[1] has NLC-width or clique-width at most k. In Section 4, we show that an incidence graph[2] has tree-width at most k if its line graph has NLC-width or clique-width at most k. In [9] it is shown that a line graph has NLC-width at most $k+2$ and clique-width at most $2k+2$ if the root graph has tree-width k. This in connection with the result of Section 4 is used to show by a reduction from tree-width minimization that minimizing NLC-width is NP-complete.

2 Preliminaries

Let $[k] := \{1, \ldots, k\}$ be the set of all integers between 1 and k. We work with finite undirected vertex labeled *graphs* $G = (V_G, E_G, \mathrm{lab}_G)$, where V_G is a finite set of *vertices* labeled by some mapping $\mathrm{lab}_G : V_G \rightarrow [k]$ and $E_G \subseteq \{\{u,v\} \mid u,v \in V_G,\ u \neq v\}$ is a finite set of *edges*. The labeled graph consisting of a single vertex labeled by $a \in [k]$ is denoted by $\bullet_a$.

The notion of clique-width is defined by Courcelle and Olariu in [7].

Definition 1 (Clique-width, [7]). *Let k be some positive integer. The class CW_k of labeled graphs is recursively defined as follows.*

1. *The single vertex graph $\bullet_a$ for some $a \in [k]$ is in CW_k.*
2. *Let $G = (V_G, E_G, lab_G) \in CW_k$ and $J = (V_J, E_J, lab_J) \in CW_k$ be two vertex disjoint labeled graphs. Then $G \oplus J := (V', E', lab')$ defined by $V' := V_G \cup V_J$, $E' := E_G \cup E_J$, and*

$$lab'(u) \ := \ \begin{cases} lab_G(u) \ \textit{if } u \in V_G \\ lab_J(u) \ \textit{if } u \in V_J \end{cases}$$

is in CW_k.

[1] The line graph $L(G)$ of a graph G has a vertex for every edge of G and an edge between two vertices if the corresponding edges of G are adjacent [20].

[2] The incidence graph $I(G)$ of a graph G is the graph we get if we replace every edge $\{u,v\}$ of G by a new vertex w and two edges $\{u,w\},\{w,v\}$.

3. *Let $a, b \in [k]$ be two distinct integers and $G = (V_G, E_G, lab_G) \in CW_k$ be a labeled graph then*
   (a) $\rho_{a \to b}(G) := (V_G, E_G, lab')$ defined by

$$lab'(u) \;:=\; \begin{cases} lab_G(u) & \text{if } lab_G(u) \neq a \\ b & \text{if } lab_G(u) = a \end{cases}$$

   is in CW_k and
   (b) $\eta_{a,b}(G) \;:=\; (V_G, E', lab_G)$ defined by

$$E' := E_G \cup \{\{u, v\} \mid u, v \in V_G,\; u \neq v,\; lab(u) = a,\; lab(v) = b\}$$

   is in CW_k.

The notion of NLC-width[3] is defined by Wanke in [19].

Definition 2 (NLC-width, [19]). *Let k be some positive integer. The class NLC_k of labeled graphs is recursively defined as follows.*

1. *The single vertex graph $\bullet_a$ for some $a \in [k]$ is in NLC_k.*
2. *Let $G = (V_G, E_G, lab_G) \in NLC_k$ and $R : [k] \to [k]$ be a function, then $\circ_R(G) := (V_G, E_G, lab')$ defined by $lab'(u) := R(lab_G(u))$ is in NLC_k.*
3. *Let $G = (V_G, E_G, lab_G) \in NLC_k$ and $J = (V_J, E_J, lab_J) \in NLC_k$ be two vertex disjoint labeled graphs and $S \subseteq [k]^2$ be a set of label pairs, then graph $G \times_S J := (V', E', lab')$ defined by $V' := V_G \cup V_J$,*

$$E' := E_G \cup E_J \cup \{\{u, v\} \mid u \in V_G,\; v \in V_J,\; (lab_G(u), lab_J(v)) \in S\},$$

   and

$$lab'(u) \;:=\; \begin{cases} lab_G(u) & \text{if } u \in V_G \\ lab_J(u) & \text{if } u \in V_J \end{cases}$$

   is in NLC_k.

The *clique-width* (*NLC-width*) of a labeled graph G is the least integer k such that $G \in \mathrm{CW}_k$ ($G \in \mathrm{NLC}_k$, respectively). An expression built with the operations $\bullet_a, \oplus, \rho_{a \to b}, \eta_{a,b}$ for integers $a, b \in [k]$ is called a *clique-width k-expression.* An expression built with the operations $\bullet_a, \times_S, \circ_R$ for $a \in [k]$, $S \subseteq [k]^2$, and $R : [k] \to [k]$ is called an *NLC-width k-expression.* Every clique-width expression (NLC-width expression) has by its recursive definition a tree structure which we call the *clique-width expression tree* (*NLC-width expression tree*, respectively). A vertex labeled graph G has *linear clique-width* (*linear NLC-width*) at most k if it can be defined by a clique-width k-expression (NLC-width k-expression, respectively) in that at least one argument of every operation $\oplus$ (every operation $\times_S$, respectively) is a single labeled vertex $\bullet_a$ [11].

The notion of tree-width and path-width is defined by Robertson and Seymour in [18] and [17], respectively.

[3] The abbreviation NLC results from the *node label controlled* embedding mechanism originally defined for graph grammars.

Definition 3 (Tree-width and path-width, [18,17]). *A* tree decomposition *of a graph $G = (V_G, E_G)$ is a pair $(\mathcal{X}, T)$ where $T = (V_T, E_T)$ is a tree and $\mathcal{X} = \{X_u \mid u \in V_T\}$ is a family of subsets $X_u \subseteq V_G$ one for each node u of T such that*

1. $\cup_{u \in V_T} X_u = V_G$,
2. *for every edge $\{v_1, v_2\} \in E_G$, there is some node $u \in V_T$ such that $v_1 \in X_u$ and $v_2 \in X_u$, and*
3. *for every vertex $v \in V_G$ the subgraph of T induced by the nodes $u \in V_T$ with $v \in X_u$ is connected.*

The width *of a tree decomposition $(\mathcal{X} = \{X_u \mid u \in V_T\}, T = (V_T, E_T))$ is $\max_{u \in V_T} |X_u| - 1$. A tree decomposition $(\mathcal{X}, T)$ is called a* path decomposition *if T is a path. The* tree-width *(*path-width*) of a graph G is the smallest integer k such that there is a tree decomposition (a path decomposition, respectively) $(\mathcal{X}, T)$ for G of width k.*

The *line graph* $L(G)$ of a graph G has a vertex for every edge of G and an edge between two vertices if the corresponding edges in G have a common vertex [20]. Graph G is called the *root graph* of $L(G)$. For any line graph with at least 4 edges the root graph is unique and can be found in linear time [15].

The *incidence graph* $I(G) = (V_{I(G)}, E_{I(G)})$ of a graph $G = (V_G, E_G)$ is the graph with vertex set $V_{I(G)} = V_G \cup E_G$ and edge set $E_{I(G)} = \{\{u, e\} \mid u \in V_G,\ e \in E_G,\ u \in e\}$. The incidence graph of G is the graph we get, if we replace every edge $\{u, v\}$ of G by a new vertex w and two edges $\{u, w\}, \{w, v\}$.

3 Line Graphs of Bounded NLC-Width

Tree-width bounded graphs can also be defined by a merging procedure of so-called *terminal graphs*, which are also called *sourced graphs*. This is a well-known property of tree-width bounded graphs, see also [2]. We will define terminal graphs with edge labels, because this will allow us to define in an easy way the edge labeled root graphs of vertex labeled line graphs.

Let k, l be two positive integers. An *l-labeled k-terminal graph* is a system

$$G = (V_G, E_G, P_G, \mathrm{lab}_G),$$

where (V_G, E_G) is a graph, $P_G = (u_1, \ldots, u_k)$ is a sequence of $k \geq 0$ distinct vertices of V_G, and $\mathrm{lab}_G : E_G \to [l]$ is an edge labeling. The vertices in sequence P_G are called *terminal vertices* or *terminals* for short. The vertex u_i, $1 \leq i \leq k$, is the i-th *terminal* of G. The other vertices in $V_G - P_G$ are called *inner vertices*. The class $\mathrm{TM}_{k,l}$ of l-labeled k-terminal graphs is recursively defined as follows.

1. The terminal graph $\overbrace{\bullet \cdots \bullet}^{r}$, $1 \leq r \leq k$, consisting of r terminals is in $\mathrm{TM}_{k,l}$.
2. The terminal graph $\bullet \overset{a}{\text{—}} \bullet$, $a \in [l]$, consisting of two terminals u, v and an edge $\{u, v\}$ labeled by a is in $\mathrm{TM}_{k,l}$ for $k \geq 2$.

3. Let $G = (V_G, E_G, P_G, \mathrm{lab}_G) \in \mathrm{TM}_{k,l}$, $P = (u_1, \ldots, u_r)$, and $f : [r] \to [r]$, be a bijection. Then the l-labeled r-terminal graph $G|^f = (V_G, E_G, P', \mathrm{lab}_G)$ with $P' = (u_{f(1)}, \ldots, u_{f(r)})$ is in $\mathrm{TM}_{k,l}$.
4. Let $G = (V_G, E_G, P_G, \mathrm{lab}_G) \in \mathrm{TM}_{k,l}$, $P = (u_1, \ldots, u_r)$, and $s \in [r]$. Integer s is also called a *decrement*. Then the l-labeled $(r-s)$-terminal graph $G|_s = (V_G, E_G, P', \mathrm{lab}_G)$ with $P' = (u_1, \ldots, u_{r-s})$ is in $\mathrm{TM}_{k,l}$.
5. Let $G = (V_G, E_G, P_G, \mathrm{lab}_G) \in \mathrm{TM}_{k,l}$ and $R : [l] \to [l]$ be a *relabeling mapping.* Then the terminal graph $\circ_R(G) = (V_G, E_G, P_G, \mathrm{lab}')$ with $\mathrm{lab}'(e) = R(\mathrm{lab}_G(e))$ for all $e \in E_G$ is in $\mathrm{TM}_{k,l}$.
6. Let $H = (V_H, E_H, P_H, \mathrm{lab}_H) \in \mathrm{TM}_{k,l}$, $J = (V_J, E_J, P_J, \mathrm{lab}_J) \in \mathrm{TM}_{k,l}$, and $|P_H| \leq |P_J|$. Then terminal graph $H \times J$ defined as follows is in $\mathrm{TM}_{k,l}$.
   (a) Take the disjoint union of $(V_H, E_H, \mathrm{lab}_H)$ and $(V_J, E_J, \mathrm{lab}_J)$, and identify the i-th terminal from H with the i-th terminal from J.
   (b) An edge e from $H \times J$ is labeled by $\mathrm{lab}_{H \times J}(e) = \mathrm{lab}_H(e)$ if it is from H and by $\mathrm{lab}_{H \times J}(e) = \mathrm{lab}_J(e)$ if it is from J.
   (c) The i-th terminal of $H \times J$ is the i-th terminal of J.
   (d) Multiple edges are eliminated by removing the corresponding edges from H.

An expression built with the operations $\overbrace{\bullet \cdots \bullet}^{r}$, $\bullet \overset{a}{—} \bullet$, $|^f$, $|_s$, $\circ_R$, and $\times$ is called a *terminal* k, l*-expression.* The terminal graph defined by a terminal k, l-expression X is denoted by $\mathrm{val}(X)$. It is easy to see that $\mathrm{TM}_{k+1,1}$ defines exactly the set of graphs of tree-width at most k, see [10].

Let $G = (V_G, E_G, P_G, \mathrm{lab}_G)$ be an edge labeled terminal graph, $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, \mathrm{lab}_\mathcal{G})$ be a vertex labeled graph, and $\pi : E_G \to V_\mathcal{G}$ be a bijection such that 1.) for every $e_1, e_2 \in E_G$, e_1 and e_2 have a common vertex if and only if $\pi(e_1)$ and $\pi(e_2)$ are adjacent in $\mathcal{G}$, and 2.) for every $e \in E_G$, $\mathrm{lab}_G(e) = \mathrm{lab}_\mathcal{G}(\pi(e))$. Then $\mathcal{G}$ is called the *labeled line graph* of G, and G is called a *labeled terminal root graph* for $\mathcal{G}$.

The next theorem shows a very tight connection between the tree-width of a graph and the NLC-width of its line graph.

Theorem 1. *If a line graph has NLC-width at most k, then its root graph has tree-width at most $4k-1$.*

Proof Sketch. Let us first observe what happens if we insert edges between two vertex labeled line graphs by an NLC-width operation. Let $G = (V_G, E_G, \mathrm{lab}_G)$ be an edge labeled graph with at least two edges. Let $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, \mathrm{lab}_\mathcal{G}) \in \mathrm{NLC}_k$ be the vertex labeled line graph of G defined by some bijection $\pi : E_G \to V_\mathcal{G}$.

Every induced subgraph of $\mathcal{G}$ defines by bijection π a unique subgraph of G in that every vertex is incident with at least one edge. Assume $\mathcal{G} = \mathcal{H} \times_S \mathcal{J}$ for some $S \subseteq [k]^2$ and two non-empty vertex labeled graphs $\mathcal{H}$ and $\mathcal{J}$. Since $\mathcal{H}$ and $\mathcal{J}$ are induced subgraphs of $\mathcal{G}$, we know that they are line graphs of two subgraphs H and J of G. Since $\mathcal{H}$ and $\mathcal{J}$ are vertex disjoint, we know that H and J are edge disjoint. Since $\mathcal{H}$ and $\mathcal{J}$ have at least one vertex, we know that H and J have at least one edge. Assume further that every pair $(a, b) \in S$ defines

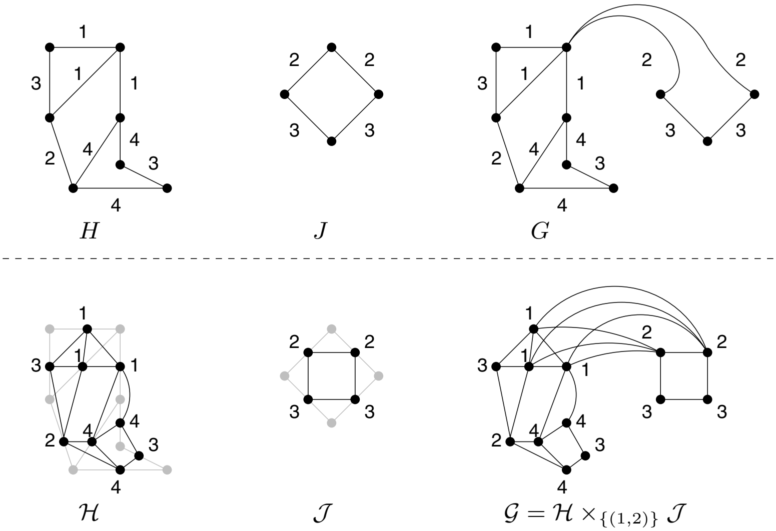

Fig. 1. An NLC-width composition $\mathcal{H} \times_{\{(1,2)\}} \mathcal{J}$ of two vertex labeled line graphs $\mathcal{H}$ and $\mathcal{J}$. The labels at the edges of H, J, and G represent the labels of the corresponding vertices of $\mathcal{H}$, $\mathcal{J}$, and $\mathcal{G}$ specified by bijection π.

at least one edge between a vertex of $\mathcal{H}$ and a vertex of $\mathcal{J}$. If S is nonempty, then in G at least one edge of H has a common vertex with at least one edge of J.

We now show that G can be defined by a vertex disjoint union of H and J and then identifying at most $4k$ vertices from H with at most $4k$ vertices from J. A simple example of such a composition $\mathcal{H} \times_S \mathcal{J}$ is shown in Figure 1.

For a label $a \in [k]$ let G_a, H_a, and J_a be the subgraphs of G, H, and J, respectively, defined by the edges e (and their end vertices) labeled by a. Let $(a, b) \in S$ be a pair of S. Then the operation $\times_S$ connects every vertex of $\mathcal{H}$ labeled by a with every vertex of $\mathcal{J}$ labeled by b. Thus, in root graph G every edge of H_a has a common vertex with every edge of J_b. Let $e = \{u, v\}$ be any edge of H_a. Then every edge of J_b either contains vertex u or vertex v. If J_b has three or more edges, then at least two of them must have a common vertex. By the same argumentation, if H_a has three or more edges then at least two of them must have a common vertex. Thus, H_a and J_b have at most two connected components. If H_a has two connected components, then all edges of every connected component have exactly one common vertex, because an edge of J_b can only contain one vertex from every of the two connected components of H_a. If H_a is connected then it contains no simple path with 6 vertices and no simple cycle with 3 or 5 vertices.

This observation leeds to a case distinction which divides all subgraphs H_a, $a \in [k]$, of H into 8 distinct types as illustrated in Figure 2. Type 8 of Figure 2 represents all graphs that have neither a vertex u such that all edges are incident with u nor two non-adjacent vertices u, v such that every edge is incident with u or v.

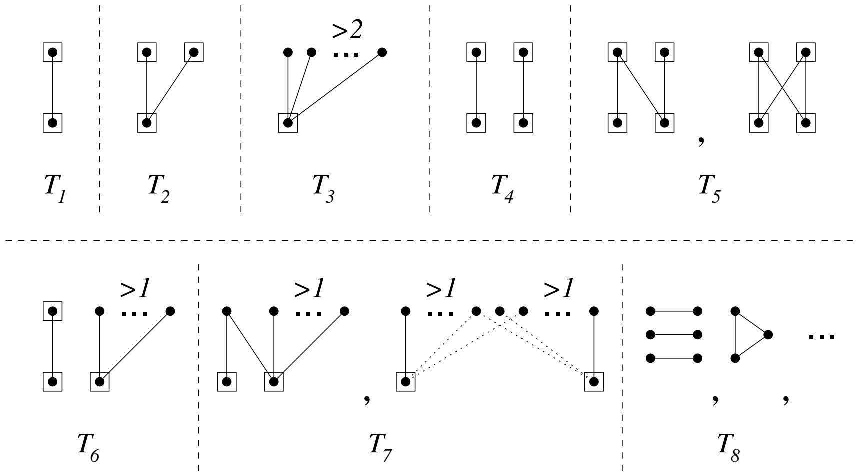

Fig. 2. Eight types for the subgraphs H_a and J_b of H and J, respectively. The specific vertices are framed by squares.

Graphs of Type 1, 2, 3, and 5 have one connected component. Graphs of Type 4 and 6 have two connected components. Graphs of Type 7 have one or two connected components. Every graph of Type 1 to 7 has at most 4 *specific* vertices of which some can be in both graphs, in H_a and in J_b. In Figure 2, these specific vertices are framed by squares.

Since the edges of G are labeled by at most k labels, it follows that at most $4k$ vertices of H are contained in J. That is, at most $4k$ vertices of H and at most $4k$ vertices of J have to be identified to define G from a vertex disjoint union of H and J. Graph G itself has also at most $4k$ vertices which can be identified with other vertices during further composition steps.

This allows us to define for an arbitrary NLC-width k-expression X that defines a line graph a mapping σ that associates for every subexpression X' of X a terminal $4k, k$-expression $\sigma(X')$ such that $\mathrm{val}(\sigma(X'))$ is the edge labeled terminal root graph of $\mathrm{val}(X')$.

1. If $X = \bullet_a$ for some $a \in [k]$ then let $\sigma(X) = \bullet \overset{a}{-} \bullet$.
2. If $X = \circ_R(X')$ for some relabeling $R : [k] \to [k]$ then let $\sigma(X) = \circ_R(\sigma(X'))$.
3. If $X = X_1 \times_S X_2$ for some $S \subseteq [k]^2$ then $\sigma(X)$ can be defined by

$$\sigma(X) = ((\sigma(X_1) \times (\sigma(X_2) \times \overbrace{\bullet \cdots \bullet}^{r})|^{f_1})|^{f_2})|_s$$

with two bijections f_1, f_2, a decrement s, and some $r \leq 4k$.
$\sigma(X)$ can be defined as above with some $r \leq 4k$, although not all terminals of $\mathrm{val}(\sigma(X_1))$ need to be identified with terminals of $\mathrm{val}(\sigma(X_2))$ via $\mathrm{val}(\overbrace{\bullet \cdots \bullet}^{r})$, or vice versa, for the complete proof of this non trivial fact see [10]. □

Since the NLC-width of a graph is always less than or equal to its clique-width [12], Theorem 1 also holds for line graphs of clique-width at most k.

Corollary 1. *If a line graph has clique-width at most k, then its root graph has tree-width at most $4k-1$.*

4 Line Graphs of Incidence Graphs

The next theorem improves the bound of Theorem 1 for line graphs of incidence graphs.

Theorem 2. *If the line graph of an incidence graph has NLC-width at most k, then its root graph has tree-width at most k.*

Proof Sketch. Let us now observe what happens if we insert edges between two vertex labeled line graphs by an NLC-width operation $\mathcal{G} = \mathcal{H} \times_S \mathcal{J}$, $S \subseteq [k]^2$ if the root graphs G, H, and J of $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{J}$, respectively, are incidence graphs. Let again G_a, $a \in [k]$, be the terminal subgraph of a terminal graph G defined by the edges (and their end vertices) labeled by a.

Since any incidence graph (and also any subgraph of an incidence graph) has no cycle of length < 6 and that every edge of an incidence graph (and also any edge of a subgraph of an incidence graph) has one end vertex of degree at most 2, every subgraph G_a, $a \in [k]$, of G can be divided into four types as illustrated in Figure 3, see [10]. Type 4 of Figure 3 represents all incidence graphs with two non-adjacent vertices u, v and an edge not incident with u or v. If G_a is of Type 4, then no vertex of G_a needs to be a terminal of G.

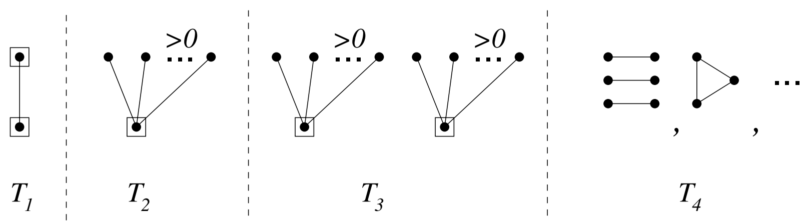

Fig. 3. Four types for the subgraphs G_a of a terminal incidence graph G. The specific vertices are framed by squares.

The same argumentation as in the proof of Theorem 1 now shows that for an arbitrary NLC-width k-expression X that defines a line graph of an incidence graph there is a mapping σ that associates for every subexpression X' of X a terminal $2k, k$-expression $\sigma(X')$ such that $\mathrm{val}(\sigma(X'))$ is the edge labeled terminal root graph of $\mathrm{val}(X')$.

We next transform $\sigma(X)$ into a terminal $2k, k$-expression Y such that every subexpression defines a connected terminal graph. This is possible, because the final root graph $\sigma(X)$ is connected, see [10]. Now every subexpression Y' of Y is of the form

1. $Y' = \bullet \overset{a}{—} \bullet$ for some $a \in [k]$,
2. $Y' = Y_1'|^f$ for some bijection f,
3. $Y' = Y_1'|_s$ for some decrement s,
4. $Y' = \circ_R(Y_1')$ for some relabeling R, or
5. $Y' = ((Y_1' \times (Y_2' \times \overbrace{\bullet \cdots \bullet}^{r})|^{f_1})|^{f_2})|_s$ for bijections f_1, f_2, some $r \leq 2k$, and a decrement s.

These subexpressions define connected terminal graphs. For every of these subexpressions Y' there is an NLC-width k-expression X' such that $\mathrm{val}(Y')$ is the edge labeled root graph of the vertex labeled line graph $\mathrm{val}(X')$.

Now we will show that Y can be transformed into an equivalent terminal $k+1, k$-expression. Let Y' be a subexpressions of Y of the form stated above and let $G = \mathrm{val}(Y')$. Let again G_a for some $a \in [k]$ be the terminal subgraph of G defined by the edges (and their end vertices) labeled by a.

1. If all subgraphs G_a, $a \in [k]$, of G are of Type 1 of Figure 3, then G has at most k edges. Since G is connected, it has at most $k+1$ terminals.
2. If all subgraphs G_a, $a \in [k]$, of G are of Type 1, 2, or 4 of Figure 3, and at least one of these subgraphs is of Type 2 or 4, then G has at least one inner vertex. In this case G has at most k terminals, see [10].
3. If some subgraph G_a, $a \in [k]$, of G is of Type 3, then two vertices u_a, v_a of G_a are terminals of G. If u_a, v_a are not adjacent in the root graph $\mathrm{val}(Y)$ we can remove them from the terminal vertex list. Otherwise we know that during any further composition these two vertices will get incident only with the missing edge $\{u_a, v_a\}$. We now modify the expression as follows.
   A subgraph of Type 3 can only be created in the following two cases.
   (a) Let
   $$G = \circ_R(H)$$
   be a graph such that G has a subgraph G_a, $a \in [k]$ of Type 3, but H has no subgraph of Type 3. Then H is connected and at least one inner vertex, and thus H has at most k terminals. We insert the edge between u_a and v_a now by
   $$G = (((\bullet \overset{a}{—} \bullet \times \circ_R (H)|^{f_1})|^{f_2})|_s)|^{f_3}$$
   with three bijections f_1, f_2, f_3 and a decrement $s = 2$. The decrement $s = 2$ removes the two vertices u_a, v_a from the terminal vertex list. (This can be done for all subgraphs G_a, $a \in [k]$, of G of Type 3 step by step.)
   (b) Let
   $$G = (H \times (J \times \overbrace{\bullet \cdots \bullet}^{r})|_{f_1})|_{f_2}$$
   be a graph such that G has a subgraph G_a of Type 3, but H and J have no subgraphs of Type 3. Then H and J are connected and have at least

one inner vertex, thus H and J have at most k terminals. Let u_a from H and v_a from J. We insert the edge between u_a, v_a of G_a by

$$G = ((H|^{f_3} \times ((J|^{f_2} \times (\bullet \overset{a}{-\!\!-} \bullet \times \overbrace{\bullet \cdots \bullet}^{r'})|^{f_1})|_{s_1} \times \overbrace{\bullet \cdots \bullet}^{r})|^{f_4})|_{s_2})|^{f_5}$$

with bijections f_1, f_2, f_3, f_4, f_5 and decrements $s_1 = 1, s_2 = 1$. If J has k' terminals then $r' = k' + 1$. Let u_a be from H and v_a be from J. One end vertex of edge $\bullet \overset{a}{-\!\!-} \bullet$ will be identified with the terminal v_a of J. Decrement $s_1 = 1$ will remove this vertex from the terminal vertex list. The other end vertex of edge $\bullet \overset{a}{-\!\!-} \bullet$ will then be identified with u_a from H. The final restriction $s_2 = 1$ will remove this vertex from the terminal vertex list. (This can be done for all subgraphs G_a, $a \in [k]$, of G of Type 3 step by step in the same way.)

In both cases, the composition step which originally inserts the edge between u_a and v_a will be omitted.

Now the resulting composition is set up with terminal graphs that have at most $k + 1$ terminals. □

Since the NLC-width of a graph is always less than or equal to its clique-width [12], Theorem 2 also holds for line graphs of incidence graphs of clique-width at most k.

Corollary 2. *If the line graph of an incidence graph has clique-width at most k, then its root graph has tree-width at most k.*

5 The NP-Completeness of NLC-Width Minimization

Since a graph G has tree-width k if and only if its incidence graph $I(G)$ has tree-width k, see for example [16], Theorem 1, 2, Corollary 1, 2 and the results of [10] together now imply the following bounds.

$$\begin{array}{llcll}
(1.) & \frac{\text{tree-width}(G)+1}{4} & \leq & \text{NLC-width}(L(G)) & \leq \text{tree-width}(G) + 2 \\
(2.) & \frac{\text{tree-width}(G)+1}{4} & \leq & \text{clique-width}(L(G)) & \leq 2 \cdot \text{tree-width}(G) + 2 \\
(3.) & \frac{\text{path-width}(G)+1}{4} & \leq & \text{linear-NLC-width}(L(G)) & \leq 2 \cdot \text{path-width}(G) \\
(4.) & \frac{\text{path-width}(G)+1}{4} & \leq & \text{linear-clique-width}(L(G)) & \leq 2 \cdot \text{path-width}(G) + 1 \\
(5.) & \text{tree-width}(G) & \leq & \text{NLC-width}(L(I(G))) & \leq \text{tree-width}(G) + 2 \\
(6.) & \text{tree-width}(G) & \leq & \text{clique-width}(L(I(G))) & \leq 2 \cdot \text{tree-width}(G) + 2 \\
(7.) & \frac{\text{path-width}(G)+1}{2} & \leq & \text{linear-NLC-width}(L(I(G))) & \leq 2 \cdot \text{path-width}(G) + 2 \\
(8.) & \frac{\text{path-width}(G)+1}{2} & \leq & \text{linear-clique-width}(L(I(G))) & \leq 2 \cdot \text{path-width}(G) + 3
\end{array}$$

Inequality (5.) can be used to show that NLC-width minimization is NP-complete.

Theorem 3. *Given a graph G and an integer k, the problem to decide whether G has NLC-width at most k is NP-complete.*

Proof. The problem to decide whether a given graph has NLC-width at most k is obviously in NP.

For a graph $G = (V, E)$ and some integer $r > 1$ let G^r be the graph G in that every vertex u is replaced by a clique C_u with r vertices and every edge $\{u, v\}$ is replaced by all edges between the vertices of C_u and C_v. That is, $G^r = (V_r, E_r)$ has vertex set $V_r = \{u_{i,j} \mid u_i \in V, j \in \{1, \ldots, r\}\}$ and edge set

$$E_r = \{\{u_{i,j}, u_{i',j'}\} \mid j, j' = 1, \ldots, r \text{ and } i = i' \vee \{u_i, u_{i'}\} \in E)\}.$$

Bodlaender et al. have shown in [3], that G has tree-width k if and only if G^r has tree-width $r(k+1)-1$.

Arnborg et al. have shown in [1] that tree-width minimization is NP-complete. That is, given a graph G and an integer k, the problem to decide whether G has tree-width at most k, is NP-complete.

For a given graph G, we first construct the graph G^3, then the incidence graph $I(G^3)$, and then the line graph $L(I(G^3))$. This can be done in polynomial time. If G has tree-width k, then G^3 has tree-width $3k+2$, and $I(G^3)$ has tree-width $3k+2$. By Theorem 2 graph $L(I(G^3))$ has NLC-width at least $3k+2$ and by Theorem 3 of [9] NLC-width at most $3k+4$. That is, tree-width$(G) = \left\lfloor \frac{\text{NLC-width}(L(I(G^3)))-2}{3} \right\rfloor$. Thus, a graph G has tree-width at most k if and only if $L(I(G))$ has NLC-width at most $3k+4$ which completes our proof. □

In [3] it is also shown that there is no polynomial time approximation algorithm for tree-width with constant difference guarantee, unless P = NP, and that for every ϵ, $0 < \epsilon < 1$, there is no polynomial time algorithm that computes for a given graph G a tree decomposition of width k such that $k -$ tree-width$(G) \leq |V_G|^\epsilon$, unless P = NP. Inequality (5.) can be used again to show similar results for NLC-width approximation, see [10].

Corollary 3.

1. *For every positive integer c there is no polynomial time approximation algorithm that computes for a given graph G an NLC-width k-expression such that $k -$ NLC-width$(G) \leq c$, unless $P = NP$.*
2. *For every ϵ, $0 < \epsilon < \frac{1}{2}$, there is no polynomial time approximation algorithm that computes for a given graph G an NLC-width k-expression such that $k -$ NLC-width$(G) \leq |V_G|^\epsilon$, unless $P = NP$.*

References

1. S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):227–284, 1987.
2. S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993.

3. H.L. Bodlaender, J.R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.
4. D.G. Corneil, M. Habib, J.M. Lanlignel, B. Reed, and U. Rotics. Polynomial time recognition of clique-width at most three graphs. In *Proceedings of Latin American Symposium on Theoretical Informatics*, volume 1776 of *LNCS*, pages 126–134. Springer-Verlag, 2000.
5. D.G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.
6. B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.
7. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000.
8. W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *Proceedings of Graph-Theoretical Concepts in Computer Science*, volume 2204 of *LNCS*, pages 117–128. Springer-Verlag, 2001.
9. F. Gurski and E. Wanke. Vertex disjoint paths on clique-width bounded graphs (Extended abstract). In *Proceedings of Latin American Symposium on Theoretical Informatics*, volume 2976 of *LNCS*, pages 119–128. Springer-Verlag, 2004.
10. F. Gurski and E. Wanke. Line graphs of bounded clique-width. Manuscript, available at "http://www.acs.uni-duesseldorf.de/∼gurski", submitted, 2005.
11. F. Gurski and E. Wanke. On the relationship between NLC-width and linear NLC-width. Manuscript, accepted for Theoretical Computer Science, 2005.
12. Ö. Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition - relationships and results for random graphs. *Congressus Numerantium*, 132:39–60, 1998.
13. Ö. Johansson. NLC_2-decomposition in polynomial time. *International Journal of Foundations of Computer Science*, 11(3):373–395, 2000.
14. D. Kobler and U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003.
15. P.G.H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *Journal of the ACM*, 21(4):569–575, 1974.
16. V. Lozin and D. Rautenbach. The tree- and clique-width of bipartite graphs in special classes. Technical Report RRR 33-2004, Rutgers University, 2004.
17. N. Robertson and P.D. Seymour. Graph minors I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35:39–61, 1983.
18. N. Robertson and P.D. Seymour. Graph minors II. Algorithmic aspects of tree width. *Journal of Algorithms*, 7:309–322, 1986.
19. E. Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54:251–266, 1994.
20. H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–168, 1932.