Linear Rank-Width and Linear Clique-Width of Trees*

Isolde Adler[1]** and Mamadou Moustapha Kanté[2]

[1] Institut für Informatik, Goethe-Universität, Frankfurt, Germany.
iadler@informatik.uni-frankfurt.de
[2] Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France.
mamadou.kante@isima.fr

Abstract We show that for every forest T the linear rank-width of T is equal to the path-width of T, and the linear clique-width of T equals the path-width of T plus two, provided that T contains a path of length three. It follows that both linear rank-width and linear clique-width of forests can be computed in linear time. Using our characterization of linear rank-width of forests, we determine the set of minimal excluded acyclic vertex-minors for the class of graphs of linear rank-width at most k.

Keywords: Linear rank-width, linear clique-width, vertex-minors

1 Introduction

Rank-width [33] is a graph parameter introduced by Oum and Seymour with the goal of efficient approximation of the *clique-width* [10] of a graph. *Linear rank-width* can be seen as the linearized variant of rank-width, similar to path-width, which can be seen as the linearized variant of tree-width. While path-width is a well-studied notion, much less is yet known about linear rank-width. Any graph of k-bounded path-width has k-bounded linear rank-width, but conversely the difference is unbounded. For example, the class of all complete (bipartite) graphs has linear rank-width at most 1, but unbounded path-width. *Linear clique-width*, a linearized version of clique-width, was introduced independently by several authors when studying the computational complexity of clique-width [13,16,17,18,19,29]. Heggernes et al. [20,21,22] investigated computing linear clique-width on restricted graph classes.

Linear rank-width is equivalent to linear clique-width in the sense that any graph class has bounded linear clique-width if and only if it has bounded linear rank-width. Computing linear rank-width is NP-complete in general. In fact, it is proved in [13] that approximating linear clique-width is NP-hard and one can easily reduce the approximation of linear clique-width to the approximation of linear rank-width. Moreover, very little is known about efficient computation of linear rank-width on restricted graph classes. The only known results are for special types of graphs, such as for complete (bipartite) graphs, and linear clique-width is known to be polynomial time computable on *thickened paths* [22] and *k-path powers* [20]. Even for the very natural class of forests efficient computability was open. In contrast, many classes are known that allow efficient computation of path-width [4,5,6,12,14,27,30,35].

In this paper, we provide the first non-trivial graph class on which linear rank-width can be computed in polynomial (even linear) time. We prove

Theorem 1 *Linear rank-width and linear clique-width of forests can be computed in linear time.*

* A preliminary version of this paper appeared in [2].

** Supported by the German Research Council, Project GalA, AD 411/1-1.

Since path-width of forests can be computed in linear time by [12], Theorem 1 is an immediate corollary of the following two theorems.

Theorem 2 *The linear rank-width of any forest equals its path-width.*

Theorem 3 *Let T be a tree. If T contains a path of length 3, then the linear clique-width of T equals the path-width of T plus 2. Otherwise, the linear clique-width of T equals the path-width of T plus 1.*

The statements of Theorems 2 and Theorem 3 fail for graphs contaning cycles. While it was known that the class of all trees has unbounded linear rank-width (see [14] for a combinatorial proof) and unbounded path-width, Theorem 2 is somewhat surprising, because it actually equates two structurally very different parameters: Path-width and linear rank-width are graph parameters based on linear orderings of the vertex set of the given graph. Intuitively, path-width seeks for an ordering that minimizes the maximum number of edges 'crossing' a cut of the ordering, while linear rank-width seeks to minimize the maximum rank of certain matrices associated with the cuts of the ordering. Linear clique-width in turn seeks to minimize the number of colours that allow to construct the given graph by introducing (coloured) vertices one by one and adding edges based on the vertex colours.

It is known that the linear clique-width of any graph is bounded by its path-width plus 2 [13]. Since linear rank-width is bounded by linear clique-width, the same bound carries over to linear rank-width. We show that the linear rank-width of any graph is bounded by its path-width. This is not hard to prove, but it seems it was not written down yet. For forests we show that the converse holds, too.

The proof of Theorem 2 uses the characterization of path-width by the cops and invisible robber game [26]. Given an ordering of the vertices of a forest T witnessing the linear rank-width of T, we construct a winning strategy for the cops. Here it is not sufficient for the cops to search the vertices according to the given ordering, but a more involved strategy yields the result. Our proof method is constructive. It shows that, given an n-vertex tree and an ordering of its vertices witnessing its linear rank-width, we can transform the given ordering into a winning strategy for the cops (using the original ordering as 'landnarks'), and hence into a path decomposition, in time $\mathcal{O}(n^2 \cdot \log^2(n))$. For determining the linear clique-width of trees we take a different approach by giving a recursive characterization of the linear clique-width of trees. The ideas we introduce here might also be useful in future research on comparing width parameters.

It is known that tree-width and path-width do not increase when taking minors. Similarly, (linear) rank-width does not increase when taking *vertex-minors* [7,23]. For a graph G and a vertex x of G, the *local complementation at* x of G consists in replacing the subgraph induced on the neighbors of x by its complement; and a graph H is a *vertex-minor* of G if H can be obtained from G by a sequence of local complementations and deletions of vertices. The set of minimal excluded vertex-minors is known to be finite for every class of graphs that is closed under vertex-minors and has bounded rank-width [32]. Moreover given k, the number of vertices of a minimal excluded vertex-minor for rank-width $\leq k$ is bounded by $(6^{k+1} - 1)/5$ [23], and hence one can theoretically compute the set of minimal exclude vertex-minors for rank-width $\leq k$ in time depending only on k by searching among all graphs of size at most $(6^{k+1} - 1)/5$ and keeping only the non-isomorphic ones which have rank-width $k+1$ and every proper vertex-minor has rank-width k. However, until now, explicit sets of minimal excluded vertex-minors are only known for circle graphs [8], distance-hereditary graphs [23], and for graphs of linear rank-width at most one [1]. For graphs of linear rank-width at most k, some minimal

excluded vertex-minors were established in [24]. Using Theorem 2, we determine the set of minimal excluded acyclic vertex-minors for linear rank-width k. It turns out that they coincide with the minimal excluded minors for graphs of path-width at most k that are acyclic [36].

Outline of the paper. Section 2 introduces the terminology, the notion of linear rank-width and the cops and invisible robber game. In Section 3 we prove that linear rank-width and path-width coincide on forests (Theorem 2), in Section 4 we prove Theorem 3 characterizing the linear clique-width of forests. In Section 5 we give the set of minimal excluded acyclic vertex-minors for the class of graphs of linear rank-width k, and we conclude with Section 6.

2 Preliminaries

For a set A we denote the power set of A by 2^A. We let $A \setminus B := \{x \in A \mid x \notin B\}$ denote the *difference* of two sets A and B. For two sets A and B let $A \Delta B := (A \setminus B) \cup (B \setminus A)$ denote the *symmetric difference* of A and B. For an integer $n > 0$ we let $[n] := \{1, \ldots, n\}$.

For sets R and C an (R, C)-*matrix* is a matrix where the rows are indexed by elements in R and columns are indexed by elements in C. (Since we are only interested in the rank of matrices, it suffices to consider matrices up to permutations of rows and columns.) For an (R, C)-matrix M, if $X \subseteq R$ and $Y \subseteq C$, we let $M[X, Y]$ be the submatrix of M obtained by restricting M to the rows and columns indexed by the elements belonging to X and Y, respectively. If M is an (R, C)-matrix and when the context is clear we will identify the row indexed by $x \in R$ with x (similarly for the column indexed by $y \in C$); hence we will say for instance that a subset X of R is a basis for the rows of M if the rows indexed by X form a basis for the rows of M and similarly for other linear algebra terminologies involving rows or columns.

In this paper, graphs are finite, simple and undirected, unless stated otherwise. Let G be a graph. We denote the vertex set of G by $V(G)$ and the edge set by $E(G)$. We regard edges as two-element subsets of $V(G)$. For a vertex $v \in V(G)$ we let $N_G(v) := \{u \in V(G) \mid \{v, u\} \in E(G)\}$ denote the set of *neighbors* of v (in G). The *degree* of v (in G) is $\deg_G(v) := |N_G(v)|$. A partition of $V(G)$ into two sets X and Y (i. e. $X \dot\cup Y = V(G)$) is called a *cut* in G, and we denote it by (X, Y). In a linear ordering $x_1, \ldots, x_n$, each $1 \leq i \leq n-1$ induces a cut (X_i, Y_i) with $X_i := \{x_1, \ldots, x_i\}$ and $Y_i := \{x_{i+1}, \ldots, x_n\}$.

A graph not containing a cycle is called *acyclic*. A *path* from u to v in a graph G is a sequence $P := u_1, \ldots, u_p$ of pairwise distinct vertices of G, where $u_0 := u_1$ and $u_p := v$, such that $\{u_i, u_{i+1}\} \in E(G)$ for every $0 \leq i \leq p-1$, and p is called the *length* of P. A *connected* graph is a graph where any two vertices are connected by a path. A *forest* is an acyclic graph and a *tree* is a connected forest. A *leaf* of a tree is a vertex of degree one. A *star* is a tree with at most one non-leaf vertex.

The *distance* between two vertices $u, v \in V(G)$ is the length of a shortest path from u to v. A *rooted tree* is a tree with a distinguished vertex r, called the *root*. The *height* of a rooted tree is the maximal length of a path from the root to a leaf. For a rooted tree T with root r and vertex $v \in V(T)$ we denote by T^v the subtree of T rooted at v and induced by those vertices $u \in V(T)$ such that the path from r to u contains v. For a rooted tree T it is sometimes convenient to orient the edges of T in the direction away from the root, thus obtaining an *oriented tree*.

Linear rank-width and path-width. Let G be a graph, and let A_G be the adjacency $(V(G), V(G))$-matrix of G. (See [15] for the basic properties of adjacency

matrices.) The *linear rank-width* of G is defined as

$$\mathrm{lrw}(G) := \min_{v_1,\ldots,v_n \text{ linear ordering of } V(G)} \max_{i\in[n-1]} \{\mathrm{rk}(A_G[X_i, Y_i])\}.$$

A *path decomposition* of G is a pair (P, B), where P is a path and $B = (B_t)_{t\in V(P)}$ is a family of subsets $B_t \subseteq V(G)$, satisfying

1. For every $v \in V(G)$ there exists a $t \in V(P)$ such that $v \in B_t$.
2. For every $e \in E(G)$ there exists a $t \in V(P)$ such that $e \subseteq B_t$.
3. For every $v \in V(G)$ the set $\{t \in V(P) \mid v \in B_t\}$ is connected in P.

The *width* of a path decomposition (P, B) is defined as $\mathrm{w}(P, B) := \max\{|B_t| \mid t \in V(P)\} - 1$. The *path-width* of G is defined as

$$\mathrm{pw}(G) := \min\{\mathrm{w}(P, B) \mid (P, B) \text{ is a path decomposition of } G\}.$$

Observation 4 *The linear rank-width (or path-width) of a graph equals the maximum of the linear rank-width (or path-width) of its connected components.*

It is easy also to see that *caterpillars*, i.e. the graphs that contain a path P such that every vertex has distance at most one to some vertex of P, have path-width ≤ 1. Indeed, the graphs of path-width at most 1 are precisely the disjoint unions of caterpillars. One can derive from [3] that graphs of linear rank-width at most 1 are exactly those that are *locally equivalent* to caterpillars (where two graphs G and H are locally equivalent, if H can be obtained from G by a sequence of local complementations). Using a labelling scheme, Ganian [14] characterizes the graphs of linear rank-width at most 1 as *thread graphs*. It is worth noticing that the path-width of trees, as well as their linear rank-width, is not bounded. For example, the rooted binary tree T_h of height h satisfies $\mathrm{pw}(T_h) = \lceil h/2 \rceil$ [34], and a combinatorial proof for the unboundedness of the linear rank-width of trees can be found in [14].

Before continuing let us first recall the following easy fact that was not written down anywhere.

Lemma 5 *Any graph G satisfies* $\mathrm{lrw}(G) \leq \mathrm{pw}(G)$.

Proof. Let (P, B) be a path decomposition of G of width $\mathrm{pw}(G)$. W. l. o. g. we may assume that (P, B) is such that any two distinct vertices $t, t' \in V(P)$ satisfy $B_t \not\subseteq B_{t'}$. Assume that the vertices of P are $t_1, \ldots, t_m$, appearing in this order on the path. Let $B'(t_1) := B(t_1)$, and for every $1 < i \leq m$ let $B'(t_i) := B(t_i) \setminus B(t_{i-1})$. Then the family $B' := (B'(t_i))_{i\in[m]}$ is a partition of $V(G)$ into non-empty sets. For each $i \in [m]$ choose an ordering of the vertices in $B'(t_i)$ and combine these orderings to an ordering of $V(G)$ that respects the path P. We claim that this ordering witnesses $\mathrm{lrw}(G) \leq \mathrm{pw}(G)$. At every cut in the ordering, the corresponding matrix has at most $\mathrm{pw}(G)$ non-zero rows. To see this, let (X, Y) be a cut in the ordering with $X = B'(t_1) \cup \ldots \cup B'(t_i) \cup X'$ and $Y = Y' \cup B'(t_{i+2}) \cup \ldots \cup B'(t_m)$, where $B'(t_{i+1}) = X' \dot\cup Y'$ is a partition of $B'(t_{i+1})$ into two sets X' and Y' with $Y' \neq \emptyset$. By the definition of path decompositions, a vertex $v \in B'(t_1) \cup \ldots \cup B'(t_i)$ is adjacent to a vertex in Y if and only if $v \in B(t_{i+1})$. Let X_0 be the set of such vertices. Now only vertices in $X_0 \cup X'$ can have neighbors in Y. Since $X_0 \cup X' \subseteq B(t_{i+1}) \setminus Y'$ and $Y' \neq \emptyset$, it follows that the rank of the matrix at (X, Y) is at most $\mathrm{pw}(G)$. □

The cops and invisible robber game. Let G be a graph and let $k \geq 0$ be an integer. The *cops and invisible robber game* on G (with game parameter k) is played by two players, the *cop player* and the *robber player*, on the graph G. The cop player controls k cops and the robber player controls the robber. Both the cops and the

robber move on the vertices of G. Some of the cops move to at most k vertices and the robber stands on a vertex r not occupied by the cops. At all times, the robber is invisible to the cops. Initially, no cops occupy vertices and the robber chooses a vertex to start playing. In each move, some of the cops fly in helicopters to at most k new vertices. During the flight, the robber sees which position the cops are approaching and before they land she quickly tries to escape by running arbitrarily fast along paths of G to a vertex r', not being allowed to run through a vertex occupied by a cop. Hence, if $X \subseteq V(G)$ is the cops' position, the robber stands on $r \in V(G) \setminus X$, and after the flight, the cops occupy the set $Y \subseteq V(G)$, then the robber can run to any vertex r' within the connected component of $G \setminus (X \cap Y)$ containing r. The cops win if they land a cop via helicopter on the vertex occupied by the robber. The robber wins if she can always elude capture.

A *play* is a sequence of cop positions $X_0, X_1, X_2, \ldots$ with $X_0 := \emptyset$ and $|X_i| \leq k$ for all i. At each step of a play, we can describe the set of *cleared* vertices as follows. At the position X_0, the set of cleared vertices is $A_0 := \emptyset$. After the cops' move to X_i (for $i > 0$), the set of cleared vertices is

$$A_i := (A_{i-1} \cup X_i) \setminus \{r \in V(G) \mid \text{there is a path from } V(G) \setminus A_{i-1} \text{ to } r \text{ in } G \setminus (X_{i-1} \cap X_i)\}.$$

Winning strategies are defined in the usual way. The *invisible cop-width* of G, $\mathrm{icw}(G)$, is the minimum number of cops having a winning strategy on G.

A winning strategy for the cops is *monotone*, if for any play $X_0, X_1, X_2, \ldots$ played according to the strategy, the sets $A_0, A_i, A_2, \ldots$ form a non-decreasing sequence (with respect to $\subseteq$). The *monotone invisible cop-width* of G, $\mathrm{monicw}(G)$, is the minimum number of cops having a monotone winning strategy on G.

Theorem 6 ([11]) *Any graph G satisfies* $\mathrm{monicw}(G) = \mathrm{pw}(G) + 1$.

3 Linear Rank-Width and Path-Width of Trees

This section is devoted to the proof of the converse direction, showing that $\mathrm{lrw}(T) \geq \mathrm{pw}(T)$ holds for any forest T. By Observation 4 we can restrict ourselves to trees instead of forests. Our proof turns a linear ordering of $V(T)$ witnessing $\mathrm{lrw}(T)$ into a winning strategy for $k+1$ cops. Roughly, the idea is that k cops move from basis to basis along the matrices at the cuts, clearing the vertices along the ordering on their way. One additional cop is used to make the transitions. For an intuition, suppose the cops occupy a vertex set B corresponding to a basis of the matrix $A_T[X_i, Y_i]$ where (X_i, Y_i) is the cut at index i. Now two cases arise, depending on whether the next vertex v_{i+1} is spanned by the rows of $A_T[X_{i+1}, Y_{i+1}]$ corresponding to B or not. The following two lemmas deal with the case that v_{i+1} is spanned by the rows of $A_T[X_{i+1}, Y_{i+1}]$ (v_{i+1} is 'dependent' on the rows of $A_T[X_{i+1}, Y_{i+1}]$). Lemma 7 highlights structural properties of the tree T in this situation, and Lemma 8 shows the next cop move in this situation (clearing vertex v_{i+1}). We start with a definition.

Let T be a tree and let (X, Y) be a cut in T. Let $B \subseteq X$ be a basis of the row space of $A_T[X, Y]$. For $x \in X \setminus B$ with $N_T(x) \cap Y \neq \emptyset$, i.e, the row vector of x in $A_T[X, Y]$ has a non-zero entry, let $B_x \subseteq B$ be the (unique) minimal subset of B spanning x. Let T_x be the subgraph of T with vertex set $V(T_x) = X' \dot\cup Y'$, where $X' := B_x \cup \{x\}$ and $Y' := N_T(B_x \cup \{x\}) \cap Y$, and with edge set $E(T') := \{\{u, v\} \in E(T) \mid u \in X',\ v \in Y'\}$. We call T_x the *B-basic tree of x (at (X, Y))*. With these assumptions we have the following two lemmas.

Lemma 7 (Spanning dependent vertices)

1. *T_x is a tree.*
2. *Each leaf z of T_x is a vertex in X', and $|N_T(z) \cap Y| = d_{T_x}(z) = 1$.*
3. *The vertices in Y' have degree two in T_x.*
4. *$|Y'| = |B_x|$.*

Proof. 1. Since T is a tree it suffices to show that T_x is connected. If not, T_x has a component C that does not contain x. Then C can be written as $C = B' \cup (N_T(B') \cap Y)$ for some non-empty subset $B' \subsetneq B_x$. Because $X' = B_x \cup \{x\}$ and B_x is the subset of B that spans the row of x in $A_T[X, Y]$, then the rows of $A_T[X', Y']$ sum up to zero, and moreover since C is a component, the rows of $A_T[X', Y']$ corresponding to B' must already sum up to zero (to see this, permute the rows and columns of $A_T[X', Y']$ in such a way, that the resulting matrix is a block matrix with non-zero blocks only along the diagonal), a contradiction to B being a basis.

2. Since the rows of $A_T[X', Y']$ sum up to zero, for every vertex $y \in Y'$, $\deg_{T_x}(y)$ is even, so y is not a leaf. Hence the leaves are in $X' \subseteq X$. Since by definition of T_x, all neighbors in Y of vertices in X' belong to T_x, the leaves of T_x are vertices in X with a unique neighbor in Y.
3. By (1), T_x is a tree. Choose x as a root and orient the edges of T_x away from the root. Recall that any vertex in Y' has even degree in T_x. Towards a contradiction, assume now that a vertex $y \in Y'$ has degree ≥ 4 (see Figure 1). Fix a successor z of y in T_x. All vertices of Y' that lie in the subtree $T_y \setminus T_z$ have even degree in $T^y \setminus T^z$. But then the sum over all those rows of $A_T[X', Y']$ that correspond to vertices in $T^y \setminus T^z$ is the zero vector, a contradiction to the fact that B is a basis. Hence every vertex in Y' has degree 2 in T_x.
4. Follows from (1) and (3). □

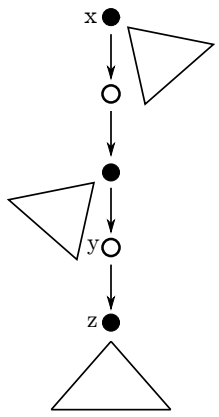

Figure 1. The tree T_x in the proof of Lemma 7. Triangles mean subtrees. Black vertices are in X', white vertices are in Y' and all white vertices have degree 2.

Lemma 8 (Clearing dependent vertices) *Suppose that in the $(k+1)$-cops and robber game on T the cops have cleared all vertices in $X \setminus \{x\}$ and the game is in a position where at most k cops are occupying vertices. Furthermore, assume that exactly $|B_x|$ cops are occupying vertices of T_x, and in addition, for each vertex $b \in B_x$, either b is occupied by a cop, or $N_T(b) \cap Y$ is occupied by cops. Then there is a sequence of monotone moves of $|B_x| + 1$ cops, involving only the cops on vertices of T_x plus one additional cop, that ends in a position, where*

1. *the vertices in* $X \cup V(T_x) \setminus \{x\}$ *are cleared,*
2. *all vertices in* Y' *are occupied,*
3. *exactly* $|B_x|$ *cops occupy vertices of* T_x*.*

Proof. Choose x as the root of T_x, and we let $g : B_x \to Y'$ where for each $z \in B_x$ we let $g(z)$ be the predecessor of z in the orientation of T_x. From Lemma 7(3) and Lemma 7(4) we know that g is a bijection. Roughly, the idea is that the cops occupying vertices of B_x move to Y' according to the bijection g. This is done bottom-up along T_x with the temporary employment of one additional cop.

By assumption, all vertices of T_x that lie in $X' \setminus \{x\} = B_x$ are cleared, and for each vertex $b \in B_x$, either b is occupied by a cop, or $N_T(b) \cap Y$ is occupied by cops. The cops make a few moves such that for all $b \in B_x$, the set $N_T(b) \cap Y$ is occupied by cops. This is done as follows. For every leaf b in T_x that is occupied by a cop, we move the cop occupying b to the unique neighbor of b in Y, i.e. to $g(b)$. For this, we place the $(k+1)$st cop on b as well, and move the (one) cop from b to the neighbor in Y. By Lemma 7(2), the leaves of T_x have a unique neighbor in Y. Now, whenever all successors of a vertex $b \in B_x \subseteq V(T_x)$ are occupied by cops, we move the cop occupying b to $g(b)$ (again with the temporary help of the $(k+1)$st cop). Proceeding like this from the leaves of T_x to the root, by Lemma 7(3), finally, exactly the vertices in Y' are occupied by cops and the vertices in $X \cup V(T_x) \setminus \{x\}$ are cleared. Part 3 follows from Lemma 7(4). This finishes the proof. □

Theorem 9 *Any tree* T *satisfies* $\mathrm{pw}(T) \leq \mathrm{lrw}(T)$*. Moreover, given* T *and a linear ordering of its vertex set* V *witnessing* $\mathrm{lrw}(T)$*, there is an algorithm of running time* $\mathcal{O}(|V|^2 \cdot \log^2(|V|))$ *that computes a path decomposition of* T *of width at most* $\mathrm{lrw}(T)$*.*

Proof. If $|V(T)| \leq 1$, then the theorem holds. Assume now that $|V(T)| \geq 2$. Let $v_1, \ldots, v_n$ be a linear ordering of $V(T)$ witnessing $k := \mathrm{lrw}(T)$. For $i \in [n]$ let $X_i := \{v_1, \ldots, v_i\}$ and $Y_i := \{v_{i+1}, \ldots, v_n\}$.

We describe a strategy for $k+1$ cops. The strategy follows the linear ordering of $V(T)$. For each new vertex v_i that has to be cleared, we describe a *transition* – a finite sequence of monotone cop moves to make sure that v_i is cleared. After the ith transition, the following invariants hold.

(I1) Every vertex in X_i is cleared.

There is a basis $B_i \subseteq X_i$ of the rows of $A_T[X_i, Y_i]$ such that

(I2) each $b \in B_i$ satisfies:

> b is occupied by a cop or $N_T(b) \cap Y_i$ is occupied by cops, and no vertex in the set $X_i \setminus B_i$ is occupied by a cop.

(I3) The cops occupy exactly $|B_i|$ vertices.

Let ℓ be the greatest index $i \leq k$ such that the rank of $A_T[X_i, Y_i]$ is equal to i. Then $\ell \geq 1$ because $|V(T)| \geq 2$. The first ℓ transitions simply consist in placing cops on the vertices $v_1, \ldots, v_\ell$, successively. Obviously, after each such transition, the invariants hold.

Suppose we have completed the ith transition, and we want to make the $(i+1)$st transition. Moving from $A_T[X_i, Y_i]$ to $A_T[X_{i+1}, Y_{i+1}]$, exactly one of the following cases occurs.

(a) In $A_T[X_{i+1}, Y_{i+1}]$, the new vertex v_{i+1} is in the span of B_i.
(b) In $A_T[X_{i+1}, Y_{i+1}]$, the new vertex v_{i+1} is not in the span of B_i.

Observe that B_i can span the rows of $A_T[X_{i+1}, Y_{i+1}]$, but may be linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$. If it is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$, then the size of a maximum linearly independent subset of B_i is $|B_i| - 1$, because deleting a column can only decrease the rank by one.

Claim 1: If the size of a maximum linearly independent subset of B_i in $A_T[X_{i+1}, Y_{i+1}]$ is $|B_i| - 1$, then there exists a vertex $v_N \in N_T(v_{i+1}) \cap B_i$ such that $B_i \setminus \{v_N\}$ is a maximum linearly independent subset of B_i in $A_T[X_{i+1}, Y_{i+1}]$.

Proof of the Claim: If B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$ and linearly independent in $A_T[X_i, Y_i]$, there exists a row of $A_T[X_i, Y_i]$ corresponding to a vertex $u \in B_i$ that is generated by $B_i \setminus \{u\}$ and that has a 1 at the column corresponding to v_{i+1}, and hence $u \in N_T(v_{i+1})$. ⊣

If B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$, we pick $v_N \in N_T(v_{i+1}) \cap B_i$ as in Claim 1 and we let $B'_i := B_i \setminus v_N$. If B_i is linearly independent in $A_T[X_{i+1}, Y_{i+1}]$, we let $B'_i := B_i$. If v_{i+1} is in the span of B_i, we let $B_{i+1} := B'_i$. Otherwise, we let $B_{i+1} := B'_i \cup \{v_{i+1}\}$. Obviously, B_{i+1} is a basis of $A_T[X_{i+1}, Y_{i+1}]$.

For each v spanned by B_{i+1} let T_v denote the B_{i+1}-basic tree of v at the cut (X_{i+1}, Y_{i+1}). The following follows from the fact that T is a tree and the vertex v_N is adjacent to v_{i+1}.

Claim 2: Assuming B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$, let $v_N \in N_T(v_{i+1}) \cap B_i$ be as in Claim 1.

(i) The B_{i+1}-basic tree of v_N does not contain v_{i+1}.
(ii) If moreover v_{i+1} is spanned by B_i in $A_T[X_{i+1}, Y_{i+1}]$, then $V(T_{v_{i+1}}) \cap V(T_{v_N}) = \emptyset$ and there is no edge other than $\{v_N, v_{i+1}\}$ between a vertex of $T_{v_{i+1}}$ and a vertex of T_{v_N}. ⊣

We distinguish two cases, depending on whether a cop occupies v_{i+1}.

Case 1. After the ith transition, v_{i+1} is not occupied by a cop.
Then by the inductive invariant (I1), the set $N_T(v_{i+1}) \cap X_{i+1} = N_T(v_{i+1}) \cap X_i$ is occupied by cops, and hence $N_T(v_{i+1}) \cap X_i \subseteq B_i$ by the inductive invariant (I2).

Case 1.1 Vertex v_{i+1} is in the span of B_i in $A_T[X_{i+1}, Y_{i+1}]$.
If v_{i+1} has no neighbors in Y_{i+1}, then we use the $(k+1)$st cop to step on v_{i+1} and remove the cop again. Otherwise, let T' be the B_{i+1}-basic tree of v_{i+1}, and let $B' \subseteq B_{i+1}$ be the minimal subset of B_{i+1} spanning v_{i+1}. Since T has no cycles, $V(T') \cap \big(N_T(v_{i+1}) \cap X_{i+1}\big) = \emptyset$. Hence we can use Lemma 8 to move to $N_T(v_{i+1}) \cap Y_{i+1}$ with at most $|B'| + 1 \leq k + 1$ cops, ending in a position where at most k cops are on $V(T)$. Since $N_T(v_{i+1}) \cap Y_{i+1}$ is occupied by cops, we can use the $(k+1)$st cop to step on v_{i+1} and then lift the $(k+1)$st cop up again, thus clearing v_{i+1}. We have then cleared X_{i+1}.

It remains to verify (I2) and (I3). By the inductive hypothesis, (I2) is already satisfied, and if B_i is linearly independent in $A_T[X_{i+1}, Y_{i+1}]$, (I3) is also satisfied. So assume B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$. If v_N does not have a neighbor in Y_{i+1} we can safely remove the cop from v_N. Otherwise, if it has a neighbor in Y_{i+1}, we can use Lemma 8 to move to $N_T(v_N) \cap Y_{i+1}$, and we then lift up the cop from v_N. By Claim 2, we can do it safely. In this way, we end the transition with a position of $|B_{i+1}|$ cops on $V(T)$. This follows from Lemma 8(3) and Claim 2. Hence all three invariants are satisfied. Finally, note that all performed cop moves are monotone.

Case 1.2. Vertex v_{i+1} is not in the span of B_i in $A_T[X_{i+1}, Y_{i+1}]$.
If B_i is linearly independent in $A_T[X_{i+1}, Y_{i+1}]$, then we place a cop in v_{i+1} and then all the three conditions are clearly satisfied. So we may assume that B_i is linearly

dependent in $A_T[X_{i+1}, Y_{i+1}]$. If v_N has no neighbors in Y_{i+1}, we place the $(k+1)$st cop on v_{i+1} (v_{i+1} is not already occupied by a cop) and we then remove the cop from v_N. After these moves, at most k cops are occupying vertices.

Now, if v_N has a neighbor in Y_{i+1}, take the B_{i+1}-basic tree T_{v_N} of v_N and use Lemma 8 to move cops in $V(T_{v_N}) \setminus \{v_N\}$ to $N_T(v_N) \cap Y_{i+1}$. Claim 2 guarantees the safety of these moves. After these moves, at most k cops are occupying vertices. If v_{i+1} was occupied by a cop, then remove the cop that is still occupying the vertex v_N. If v_{i+1} was not occupied by a cop, then we place the $(k+1)$st cop on v_{i+1} and remove the cop that occupy the vertex v_N. After these moves, v_{i+1} is cleared and since we did not recontaminate X_i, X_{i+1} is cleared. Moreover, exactly $|B_{i+1}|$ vertices of T are occupied by cops (Lemma 8(3) and Claim 2), and since the other cops are not moved, (I2) is satisfied. Hence the three invariants are satisfied. Again, note that all performed cop moves are monotone.

Case 2. After the ith transition, v_{i+1} is occupied by a cop.
By the inductive invariant (I1), each vertex $b \in N_T(v_{i+1}) \cap X_{i+1} = N_T(v_{i+1}) \cap X_i$ is cleared, hence either b is occupied by a cop, or $N_T(b) \cap Y_{i+1}$ is occupied by cops.

Case 2.1. Vertex v_{i+1} is in the span of B_i in $A_T[X_{i+1}, Y_{i+1}]$.
For every $b \in \{v_N, v_{i+1}\}$ such that $V(T_b) \cap Y_{i+1}$ contains an unoccupied vertex, we use Lemma 8 to move cops in $V(T_b) \setminus \{b\}$ to $V(T_b) \cap Y_{i+1}$. This is possible, because the B_{i+1}-basic trees involved are pairwise disjoint and pairwise connected via v_{i+1} only (Claim 2). After that, we remove the cops occupying vertices in $\{v_N, v_{i+1}\}$. The cop moves are monotone, and we can conclude that the three inductive invariants are satisfied.

Case 2.2. Vertex v_{i+1} is not in the span of B_i in $A_T[X_{i+1}, Y_{i+1}]$.
As in Case 1.2 we may assume that B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$, otherwise the three invariants are trivially satisfied. If $V(T_{v_N}) \cap Y_{i+1}$ contains an unoccupied vertex, we use Lemma 8 to move cops in $V(T_{v_N}) \setminus \{v_N\}$ to $V(T_{v_N}) \cap Y_{i+1}$. After that, we remove the cop occupying the vertex v_N. The cop moves are monotone, and we can conclude that the three inductive invariants are satisfied.

Let us now discuss the time complexity and assume T has n vertices. From **Cases 1** and **Cases 2** in order to clean the vertex v_{i+1} we may move some cops to $N_T(v_{i+1}) \cap Y_{i+1}$ and/or to $N_T(v_N) \cap Y_{i+1}$, and this can be done in time $O(n)$ using the B_{i+1}-basic trees of v_{i+1} and v_N. So it is enough to show that the B_{i+1}-basic trees of v_N and v_{i+1} can be constructed in time $O(n \cdot \log^2(n))$. If $|B_i| = k$, then by Gaussian elimination one can compute a row basis B_{i+1} of $A_T[X_{i+1}, Y_{i+1}]$ in time $O(k^2 \cdot n)$, and eventually identify the subset S of B_{i+1} that generates v_N if B_i is linearly dependent in $A_T[X_{i+1}, Y_{i+1}]$, and similarly we can identify the subset S' of B_{i+1} that generates v_{i+1} if this latter one is in the span of B_i. Now, from S and S' we can compute the B_{i+1}-basic trees of v_N and of v_{i+1}, respectively, in time $O(n)$. Since $k \leq \log(n)$ (the linear rank-width of a tree is bounded by $\log(n)$), we can conclude that v_{i+1} can be cleaned in time $O(n \cdot \log^2(n))$, and hence a winning strategy with $k+1$ cops can be computed in time $\mathcal{O}(n^2 \cdot \log^2(n))$, which in turn gives rise to a path decomposition of width k. □

Note that the statements of Theorems 2 and Theorem 3 fail for K_n, the complete graph with $n \geq 3$ vertices. While $\mathrm{lrw}(K_n) = 1$ and $\mathrm{lcw}(K_n) = 2$, we have $\mathrm{pw}(K_n) = n - 1$.

Proof of Theorem 2. The theorem now follows from Lemma 5 and Theorem 9. □

Theorem 2 combined with [12] immediately gives the following as a corollary.

Corollary 10 *There is a linear time algorithm that computes the linear rank-width of any forest, and an ordering of its vertex set V witnessing its linear rank-width can be computed in time $\mathcal{O}(|V| \cdot \log(|V|))$.*

We conclude this section with two examples illustrating the cop strategy in the proof of Theorem 9. In particular, the first one shows that it is not sufficient for the cops to simply follow the vertices along the linear ordering witnessing the linear rank-width.

Example 11 *1. Let T be the graph shown in Figure 2(a). The ordering b, a, c, d, e is a witness for* $\mathrm{lrw}(T) \leq 1$. *The strategy for two cops according to the proof of Theorem 9 is as follows: the first cop moves to b and then the second cop moves to a and remains there. Now the first cop moves to c, d, e in this order.*

2. The tree T in Figure 2(b) satisfies $\mathrm{lrw}(T) = 2$. *The given ordering (attached to the vertices) witnesses* $\mathrm{lrw}(T) \leq 2$. *The strategy for three cops according to Theorem 9 is* $\{1\}, \{1,2\}, \{2,3\}, \{2,4\}, \{4,5\}, \{4,5,6\}, \{4,6,7\}, \{4,8\}, \{8,9\}, \{8,9,10\}, \{8,10,11\}, \{8,10,12\}, \{8,12,13\}, \{8,13,14\}, \{8,14,15\}, \{8,16\}, \{8,16,17\}, \{8,17,18\}, \{8,17,19\}, \{8,19,20\}, \{19,20,21\}, \{21,22\}$.

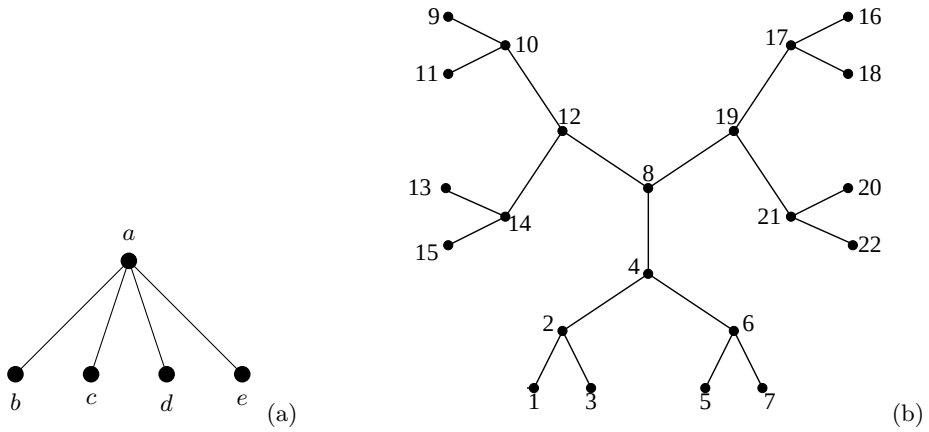

Figure 2. The star of Example 11(1) and the tree of Example 11(2).

4 Linear Clique-Width and Path-Width

In this section we prove Theorem 3, characterizing the linear clique-width of trees in terms of their path-width.

Let us recall the definition of linear clique-width [13,18,29]. Let k be a positive integer. A *k-labeled* graph is a pair (G, γ) where G is a graph and $\gamma : V(G) \to [k]$ is a mapping (that maps each vertex to one of k *labels*); we will also denote it by $(V(G), E(G), \gamma)$. The k-labeled graph consisting of a single vertex labeled by $i \in [k]$ is denoted by $(\mathbf{i}, \gamma_{\mathbf{i}})$. The set LIN-CW$_k$ of k-labeled graphs is defined inductively with the following operations.

1. For each $i \in [k]$, $(\mathbf{i}, \gamma_{\mathbf{i}})$ is in LIN-CW$_k$.
2. If $i, j \in [k]$ and (G, γ) is in LIN-CW$_k$, then $(\rho_{i \to j}(G), \gamma')$ is in LIN-CW$_k$ and denotes the k-labeled graph $(V(G), E(G), \gamma')$ with

$$\gamma'(x) := \begin{cases} \gamma(x) & \text{if } \gamma(x) \neq i, \\ j & \text{if } \gamma(x) = i. \end{cases}$$

3. If $i, j \in [k]$, $i \neq j$, and (G, γ) is in LIN-CW$_k$, then $(\eta_{i,j}(G), \gamma)$ is in LIN-CW$_k$ and denotes the k-labeled graph $(V(G), E', \gamma)$ with

$$E' := E(G) \cup \{\{x, y\} \mid \gamma(x) = i \text{ and } \gamma(y) = j\}.$$

4. If $i \in [k]$ and (G, γ) is in LIN-CW$_k$, then $(G \oplus \mathbf{i}, \gamma')$ is in LIN-CW$_k$ and denotes the graph $(V(G) \cup \{z\}, E(G), \gamma')$ where $z \notin V(G)$ and

$$\gamma'(x) := \begin{cases} \gamma(x) & \text{if } x \in V(G), \\ i & \text{if } x = z. \end{cases}$$

An expression built with the operations $\mathbf{i}, \rho_{i\to j}, \eta_{i,j}$ and $\oplus$ according to the definition of LIN-CW$_k$ is called a *linear k-expression*. The *linear clique-width* of a graph G, denoted by $\mathrm{lcw}(G)$, is the minimum k such that G is isomorphic to a graph in LIN-CW$_k$ (after forgetting the labels). For example the linear clique-width of a path of length 3 is 3. Note that if H is an induced subgraph of G, then $\mathrm{lcw}(H) \leq \mathrm{lcw}(G)$. Moreover, any linear k-expression t defining a graph G defines a linear ordering of $V(G)$ witnessing the ordering in which the vertices of G appear in t.

Lemma 12 ([9,13]) *Any graph G satisfies* $\mathrm{lcw}(G) \leq \mathrm{pw}(G) + 2$. □

Lemma 13 *Let T be a tree obtained from three trees T_1, T_2 and T_3 by adding a new vertex r adjacent to one vertex in each of the three trees. If* $\mathrm{lcw}(T_i) \geq k$ *for each* $i \in \{1, 2, 3\}$, *then* $\mathrm{lcw}(T) \geq k + 1$.

Proof. If $k = 1$ then each T_i is a single vertex and hence T is a star with 3 leaves, and we have $\mathrm{lcw}(T) = 2$, because at least two different labels are necessary for adding an edge.

Assume that $k > 1$. This implies in particular, that each T_i has at least two vertices. Let $\ell := \mathrm{lcw}(T)$. We show that $\ell \geq k + 1$. Let t be a linear ℓ-expression defining T. and let $\pi := (v_1, \ldots, v_n)$ be the linear ordering of $V(T)$ corresponding to t. W. l. o. g. assume that the operations in t are carried out in such a way, that each introduction of a new vertex v (operation type 1) is followed by a disjoint union (operation type 4), which in turn is followed by all possible edge additions involving v (operations of type 3) and finally by recolorings that leave us with the smallest number of labels that is possible at this step [9, Proposition 2.101]. Moreover, we can assume that there is one distinguished label, the *unlabel*, that is assigned to a vertex once all its neighbors in T have been constructed (see for instance [31]).

Assume that $v_1 \notin V(T_2)$ and $v_n \notin V(T_2)$. Let $\pi_2 := u_1, \ldots, u_p$ be the linear ordering of $V(T_2)$ obtained by restricting π to $V(T_2)$ and let t_2 be the corresponding subterm of t, which by the assumption above is a linear k-expression defining the tree T_2. Let (X, Y) be a cut in π_2, where X is partitioned into at least k label classes (defined by the pre-images of the $\geq k$ different labels), and assume that in t_2 we cannot reduce the number of labels at (X, Y) by a recoloring operation. Such a cut exists because by assumption, $\mathrm{lcw}(T_2) = k$. If X is partitioned into more than k label classes, then we are done, because then $\ell \geq k + 1$. Hence assume that X is partitioned into exactly k label classes.

Choose a cut (X', Y') of π such that $X \subseteq X'$ and $Y \subseteq Y'$. Since $T \setminus T_2$ is connected and by the choice of t_2, there are two vertices x and y such that $x \in X' \setminus X$, $y \in Y' \setminus Y$, and $\{x, y\} \in E(T)$. Assume w.l.o.g that $x \in V(T_1)$. If y has no neighbor in X, then, in t, the vertex x cannot have the same label as any other vertex in X, and, since X is already partitioned into k label classes, this shows that $\ell \geq k + 1$.

Assume now that there is a vertex z in X that is a neighbor of y. Then $y = r$, $\{z\} = N_T(y) \cap V(T_2)$ and similarly $\{x\} = N_T(y) \cap V(T_1)$ because by the assumption

r has only one neighbor in T_2 and also one neighbor in T_1. Now if one of the neighbors of z in T_2 is contained in Y, then x cannot have the same label as any other vertex in X, and hence $\ell \geq k+1$. Hence we may assume that $N_{T_2}(z) \subseteq X$. If x and z have different labels, then the label of x must be unique and $\ell \geq k+1$. Hence we may assume that x and z have the same label. Then no other vertex in X has this label.

Towards a contradiction, suppose that at the cut (X', Y') in t there are only k labels. If there is a vertex x' in T_1 with $\{x'\} \cup N_T(x') \subseteq X'$, then by the assumption above x' is labelled by the *unlabel* in t and we can reduce the number of labels at the cut (X, Y) in t_2 by assigning the *unlabel* to z (because z does not have the same label as any other vertex in X) – a contradiction to the choice of (X, Y). Hence every vertex in $V(T_1) \cap X'$ has a neighbor in Y'. Since x and z have the same label, all the neighbors of $N_{T_1}(x) \subseteq X'$. Using this and the fact that $k \geq 2$, we see that there exists at least one vertex $x' \neq x$ in $V(T_1) \cap X'$ and this vertex must have a neighbor in Y'. But no vertex in T_2 can have the same label as x'. Again, we can reduce the number of labels at the cut (X, Y) in t_2 by assigning the label of x' to z – a contradiction to the choice of (X, Y). Hence there are at least $k+1$ labels at the cut (X', Y') in t, proving that $\mathrm{lrw}(T) \geq k+1$. □

We use the following Lemma, proved in [12, Theorem 3.1].

Lemma 14 ([12]) *Let T be a tree and let $k \geq 1$ be an integer. Then $\mathrm{pw}(T) \leq k$ if and only if for all $v \in V(T)$ at most two of the trees in $T \setminus v$ have path-width k and all others have path-width less than k.* □

Lemma 15 *Any tree T containing a path of length three satisfies $\mathrm{lcw}(T) \geq \mathrm{pw}(T) + 2$.*

Proof. We use induction on $k := \mathrm{pw}(T)$. If $k = 1$ we are done because the linear clique-width of paths of length three is 3. If $k = 2$, then any tree of path-width 2 contains the tree R_3 obtained from the star with 3 leaves by subdividing once each edge (cf. Figure 3), and by [21] $\mathrm{lcw}(R_3) = 4$. Now assume that for some $k \geq 2$, any tree having path-width $\ell \leq k$ and containing a path of length three has linear clique-width at least $\ell + 2$. Let T be a tree that contains a path of length three and satisfies $\mathrm{pw}(T) = k+1$. By Lemma 14 there exists a vertex $r \in V(T)$ such that at least three trees in $T \setminus r$ have path-width at least k. Since $k \geq 2$ each of these trees contain a path of length at least three, and by induction, these trees have linear clique-width at least $k+2$ and hence by Lemma 13, $\mathrm{lcw}(T) \geq k+3$. □

Proof of Theorem 3. The first statement follows from Lemmas 12 and 15. For the second statement, if T does not contain a path of length three, then it is a star. Since stars with at least one edge have linear clique-width 2 and path-width 1, we can conclude that $\mathrm{lcw}(T) = \mathrm{pw}(T) + 1$. □

Before extending the result to forests, let's warm up with some examples. The forest T consisting of two isolated edges satisfies $\mathrm{lcw}(T) = 3$, while each of the connected components of T has linear clique-width 2. Moreover, the forest T' consisting of an isolated vertex and an isolated edge satisfies $\mathrm{lcw}(T') = 2$ (first construct the edge), and the forest T'' obtained from T' by adding a second isolated edge satisfies $\mathrm{lcw}(T'') = 3$. For extending our results from trees to forests, we use the following straightforward observation.

Observation 16 *Let T be a forest with at least one edge. If T contains a path of length 3, then $\mathrm{lcw}(T)$ is equal to the maximum of the linear clique-widths of its connected components. If T contains no path of length 3, and exactly one connected component of T assumes the maximum of the linear clique-widths of the connected components of T, then $\mathrm{lcw}(T)$ is equal to the maximum of the linear clique-widths of its connected components. If T contains no path of length 3 and at least two of*

the connected components of T assume the maximum of the linear clique-widths of the connected components of T, then $\mathrm{lcw}(T)$ *is equal to the maximum of the linear clique-widths of its connected components* $+1$.

Proof. Let $T_1, T_2, \ldots, T_p$ be the connected components of T, and let $k := \max\{\mathrm{lcw}(T_i)\}$. Let $t_1, t_2, \ldots, t_p$ be respectively the linear expressions defining $T_1, T_2, \ldots, T_p$, and let $t'_i := \bigcirc_{\rho_{i\to 0}}(t_i)$, where 0 is the *unlabel* color, that is never used to create an edge. Let $t := t_p \oplus t'_{p-1} \oplus \cdots \oplus t'_1$. It is clear that t defines T because the color 0 is never used to create an edge. Assume T contains a path of length 3 and let $T_1, \ldots, T_\ell$ be the trees with a path of length 3. Then, for each $1 \leq i \leq \ell$, $\mathrm{lcw}(T_i) = \mathrm{pw}(T_i)+2$ and t_i uses surely the *unlabel* color, and for $\ell+1 \leq i \leq p$, we have $\mathrm{lcw}(T_i) = \mathrm{pw}(T_i)+1$. Therefore, the linear expression t uses k colors.

If any of the T_i's has a path of length 3, then T is a disjoint union of stars and $k = 2$. Now, if $\mathrm{lcw}(T_1) = 2$, and $\mathrm{lcw}(T_i) \leq 1$ for all $i \neq 1$, then $\bigoplus_{2\leq i\leq p} \mathbf{1} \oplus (\eta_{1,2}(\mathbf{1} \oplus \mathbf{2}))$ clearly defines T. Finally, if $\mathrm{lcw}(T_1) = \mathrm{lcw}(T_2) = k = 2$, then it is proved in [31] that the linear clique-width of T is 3. □

Corollary 1. *There is a linear time algorithm that computes the linear clique-width of any n-vertex forest T, and a linear* $\mathrm{lcw}(T)$*-expression can be computed in time* $\mathcal{O}(n \cdot \log(n))$.

Proof. From [12] there is a linear time algorithm that computes the path-width of any forest, and an optimal path-decomposition can be computed in time $\mathcal{O}(n \cdot \log(n))$ for every n-vertex forest. Hence, the statement follows from Observation 16 characterizing the linear clique-width of forests, and the proof of Lemma 12 in [9], which computes from an optimal path-decomposition (P, B) of width k a linear $(k+2)$-expression in time $O(\max\{k, \ell\} \cdot n)$ where ℓ is the maximum number of edges in a bag of (P, B). □

5 Minimal Excluded Acyclic Vertex-Minors

As an application, in this section we identify the minimal excluded *acyclic* vertex-minors for linear rank-width k by using both Lemma 14 and Theorem 2.

For a graph G and a vertex x of G, the *local complementation at x* of G consists in replacing the subgraph induced on the neighbors of x by its complement. The resulting graph is denoted by $G * x$. If H can be obtained from G by a sequence of local complementations, then G and H are called *locally equivalent*. A graph H is called a *vertex-minor* of a graph G if H is isomorphic to a graph obtained from G by applying a sequence of local complementations and deletions of vertices. The graph H is a *proper vertex-minor* of G if H is a vertex-minor of G and $|V(H)| < |V(G)|$. A graph G is a *minimal excluded vertex-minor* for the class of graphs of linear rank-width k, if $\mathrm{lrw}(G) > k$ and $\mathrm{lrw}(H) \leq k$ for all proper vertex-minors H of G. See [7,23] for more information on vertex-minors.

We say that a graph G is a *minimal excluded acyclic vertex-minor* for the class of graphs of linear rank-width k, if G is acyclic and every proper acyclic vertex-minor of G has linear rank-width less than k. Note that a minimal excluded acyclic vertex-minor may not be a minimal excluded vertex-minor. For example, R_3 of Figure 3 is a minimal excluded acyclic vertex-minor for the class of graphs of linear rank-width at most 1, but it contains the net graph, also of Figure 3, as a proper vertex-minor, which in turn is a minimal excluded vertex-minor for the class of graphs of linear rank-width at most one [1].

We now determine the set of pairwise not locally equivalent minimal excluded acyclic vertex-minors for linear rank-width k. Due to minimality, the minimal excluded (acyclic) vertex-minors for linear rank-width k are necessarily connected.

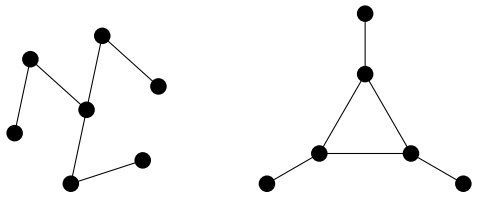

Figure 3. The subdivided 3-star R_3, and the net graph.

Let $\mathcal{H}_1 := \{R_3\}$. For $k \geq 2$, let $\mathcal{H}_k$ be the set of (pairwise non isomorphic) trees obtained by taking a new vertex r and three trees in $\mathcal{H}_{k-1}$, and by linking this new vertex to one vertex in each of these three trees. Notice that all the trees in $\mathcal{H}_k$ have the same size. By $\mathcal{R}_k$ we denote the set of (pairwise non isomorphic) trees T' obtained from trees $T \in \mathcal{H}_k$ by adding a new vertex adjacent to one vertex of T. A *rooted tree in* $\mathcal{H}_k \cup \mathcal{R}_k$ is a tree T in $\mathcal{H}_k \cup \mathcal{R}_k$ rooted at some vertex.

A rooted tree H is a *rooted vertex-minor* of a rooted tree G if H is a vertex-minor of G, H is obtained without applying a local complementation at the root of G and the root of H is mapped to the root of G. We recall that if T is a rooted tree and v is a vertex of T then we denote by T^v the subtree of T rooted at v.

Lemma 17 *Let $k \geq 2$ and let T be a rooted tree of linear rank-width k. Let $2 \leq \ell \leq k$ and let $v \in V(T)$ be such that T^v has linear rank-width ℓ. Then T^v has a rooted tree in $\mathcal{H}_{\ell-1} \cup \mathcal{R}_{\ell-1}$ as a rooted vertex-minor.*

Proof. We prove it by induction on ℓ. Let v be such that T^v has linear rank-width 2. By Lemma 14 there exists a vertex u in T^v such that T^u is isomorphic to a rooted star, rooted at its center, by subdividing at least once three of its edges. Let H be the rooted subtree of T^v induced by the vertices in the path from v to u and three paths of length two originated from u. It is clear that H^u is isomorphic to a rooted tree in $\mathcal{H}_1$ (by taking in the tree in $\mathcal{H}_1$ the vertex of degree three as root). Now it is clear that H admits a rooted tree in $\mathcal{H}_1 \cup \mathcal{R}_1$ as a rooted vertex-minor by applying local complementations at intermediate vertices in the path from v to u and removing them after each local complementation. Since H is a rooted vertex-minor of T^v, we are done.

Now assume the claim is true for all vertices v of T such that T^v has linear rank-width at most ℓ' and let u be a vertex of T such that T^u has linear rank-width $\ell' + 1$. By Lemma 14 there exists a vertex v in T^u such that T^v has linear rank-width $\ell' + 1$ and three neighbors v_1, v_2 and v_3 such that T^{v_i} has linear rank-width ℓ'. By inductive hypothesis for each $i \in \{1, 2, 3\}$ there exists a rooted tree H_i in $\mathcal{H}_{\ell'-1} \cup \mathcal{R}_{\ell'-1}$ such that H_i is a rooted vertex-minor of T^{v_i}. For each $i \in \{1, 2, 3\}$ the local complementations applied to obtain H_i do not modify the tree in $T \setminus T^{v_i}$. So the rooted tree obtained from T^v and composed of the H_is and of the paths from u to the v_is is a rooted vertex-minor of T^u. We obtain a tree in $\mathcal{H}_{\ell'} \cup \mathcal{R}_{\ell'}$ as a rooted vertex-minor of T^u as follows. If H_i is in $\mathcal{R}_{\ell'-1}$, then apply a local complementation at v_i, and then remove v_i. In this way, we get a vertex-minor of T^v rooted at v and isomorphic to a rooted tree in $\mathcal{H}_{\ell'}$. By applying local complementations at intermediate vertices in the path from v to u and removing them after each local complementation we get a rooted tree in $\mathcal{H}_{\ell'} \cup \mathcal{R}_{\ell'}$ as a rooted vertex-minor of T^u. □

Lemma 18 *Let $k \geq 1$ be an integer. Every tree of linear rank-width $k+1$ contains a tree in $\mathcal{H}_k$ as a vertex-minor.*

Proof. Let T be a tree with linear rank-width $k+1$. By Theorem 2 and Lemma 14 there exists a vertex r and three trees T_1, T_2 and T_3 of $T \setminus r$ such that each T_i has linear rank-width k and has a vertex r_i adjacent to r. Let $F := T[V(T_1) \cup V(T_2) \cup V(T_3) \cup \{r\}]$, which is a subtree of T. Notice that by Lemma 14 F has linear rank-width $k+1$. Let us root F in r and let r_1, r_2 and r_3 denote the neighbors of r in F. By Lemma 17 each T_i has a rooted tree H_i in $\mathcal{H}_{k-1} \cup \mathcal{R}_{k-1}$ as a rooted vertex-minor. Then the tree F' composed of the rooted trees H_is and of the edges $\{r, r_i\}$ is a rooted vertex-minor of F. We obtain a tree in $\mathcal{H}_k$ from F' as follows. If H_i is in $\mathcal{R}_{k-1}$, then apply a local complementation at r_i, and then remove r_i. Since we can do each local complementation independently, we are done. □

Theorem 19 *For each $k \geq 1$, the set $\mathcal{H}_k$ is the set of minimal excluded acyclic vertex-minors for linear rank-width k.*

Proof. One can prove by induction, by using Theorem 2 and Lemma 14, that each tree in $\mathcal{H}_k$ has linear rank-width $k+1$ and is minimal (as a tree) with respect to this property. Moreover, by Lemma 18 any tree of linear rank-width $k+1$ contains as a vertex-minor a tree in $\mathcal{H}_k$. So it is enough to prove that two trees in $\mathcal{H}_k$ are not locally equivalent. Bouchet has proved in [7] that two trees are locally equivalent if and only if they are isomorphic. Hence, since no two trees in $\mathcal{H}_k$ are isomorphic, we are done. □

Notice that Theorem 19 is another unexpected relation between linear rank-width and path-width of trees. In fact, as proved in [36] the set $\mathcal{H}_k$ is also the set of acyclic (topological) minor obstructions for path-width k.

6 Conclusion

We proved that linear rank-width and path-width coincide on forests, and we determined the linear clique-width of forests in terms of their path-width. While the proof of the former result uses a game characterization of path-width, the proof of the latter is based on a Lemma 13 which gives a recursive characterization of the linear clique-width of trees (similar to Lemma 14 for path-width of trees). Indeed, linear rank-width can be characterized recursively in a similar way [25, Lemma 4.1], and this characterization can be used to prove the equality of linear rank-width and path-width on forests. Our characterizations imply the existence of linear time algorithms for computing the linear rank-width and the linear clique-width of forests. The recent paper [3] gives a polynomial time algorithm for computing the linear rank-width of distance-hereditary graphs. It is still wide open, whether a similar result can be obtained for computing linear clique-width. More generally, it is natural to ask whether there is a polynomial time algorithm for computing linear rank-width (or path-width or linear clique-width) on graphs of bounded rank-width. It is also open, whether there is a polynomial time algorithm that computes linear rank-width (or linear clique-width) on series-parallel graphs.

It should be possible to extend our methods to characterize linear NLC-width (cf. [18]) of trees in terms of their path-width (similar to Theorem 3), but it seems harder to identify the exceptions (as the stars for clique-width) and the basic graphs to start the induction (as the R_3 graph).

We used the fact that linear rank-width and path-width coincide on forests to determine the set of minimal excluded acyclic vertex-minors for linear rank-width k. One can probably use the same technique to compute the set of *minimal excluded acyclic induced subgraphs* for linear rank-width and linear clique-width k. The complete set of minimal excluded vertex-minors for linear rank-width k is unknown and a next step could be to determine the set of distance-hereditary

excluded vertex-minors for linear rank-width k (we know from [24] and [3] that their number is doubly exponential in k). In [23] it is proved that the size of the excluded vertex-minors for rank-width k is bounded by $(6^{k+1} - 6)/5$, and similar results exist for tree-width and path-width [28]. Can we get a similar result for linear rank-width? Such a bound would prove the existence of an effective algorithm for checking whether a graph has linear rank-width at most k, for fixed k, while we only know from [32] that an algorithm exists.

Contrary to (linear) rank-width (linear) clique-width is not monotone with respect to vertex-minor inclusion. For example, the path P on three edges has (linear) clique-width 3, while the graph G obtained from P by a local complementation at a vertex of degree 2 in P is a triangle with a pendant edge, and G has (linear) clique-width 2. Characterizing linear clique-width with respect to the induced subgraph inclusion seems to be a hard task and few results have been obtained [16,22]. Can we at least characterize the linear clique-width of co-graphs (which have clique-width at most 2) or, more generally, of distance-hereditary graphs (which have clique-width at most 3) in order to identify the set of distance-hereditary excluded induced subgraphs for linear clique-width k?

The characterizations of linear rank-width and linear clique-width of forests in terms of their path-width are surprising. However, we believe that forests are exceptional and, except for special types of graphs (e. g. grids), such characterizations are not to be expected. But we believe that the results in this paper can be used to (approximately) compute the linear rank-width or clique-width of tree-like graphs.

References

1. Isolde Adler, Arthur M. Farley, and Andrzej Proskurowski. Obstructions for linear rank-width at most 1. *Discrete Applied Mathematics*, 168:3–13, 2014.
2. Isolde Adler and Mamadou Moustapha Kanté. Linear rank-width and linear clique-width of trees. In Andreas Brandstädt, Klaus Jansen, and Rüdiger Reischuk, editors, *WG*, volume 8165 of *Lecture Notes in Computer Science*, pages 12–25. Springer, 2013.
3. Isolde Adler, Mamadou Moustapha Kanté, and O.-joung Kwon. Linear rank-width of distance-hereditary graphs. In Dieter Kratsch and Ioan Todinca, editors, *Graph-Theoretic Concepts in Computer Science - 40th International Workshop, WG 2014. Revised Selected Papers*, volume 8747 of *Lecture Notes in Computer Science*, pages 42–55. Springer, 2014.
4. Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.
5. Hans L. Bodlaender, Ton Kloks, and Dieter Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM J. Discrete Math.*, 8(4):606–616, 1995.
6. Hans L. Bodlaender and Rolf H. Möhring. The pathwidth and treewidth of cographs. In John R. Gilbert and Rolf G. Karlsson, editors, *SWAT*, volume 447 of *Lecture Notes in Computer Science*, pages 301–309. Springer, 1990.
7. André Bouchet. Transforming trees by successive local complementations. *J. Graph Theory*, 12(2):195–207, 1988.
8. André Bouchet. Circle graph obstructions. *J. Comb. Theory, Ser. B*, 60(1):107–144, 1994.
9. Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic, A Language-Theoretic Approach*. Cambridge University Press, 2012.
10. Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.
11. Nick D. Dendris, Lefteris M. Kirousis, and Dimitrios M. Thilikos. Fugitive-search games on graphs and related parameters. *Theor. Comput. Sci.*, 172(1-2):233–254, 1997.
12. Jonathan A. Ellis, Ivan Hal Sudborough, and Jonathan S. Turner. The vertex separation and search number of a graph. *Inf. Comput.*, 113(1):50–79, 1994.

13. Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is np-complete. *SIAM J. Discrete Math.*, 23(2):909–939, 2009.
14. Robert Ganian. Thread graphs, linear rank-width and their algorithmic applications. In Costas S. Iliopoulos and William F. Smyth, editors, *IWOCA*, volume 6460 of *Lecture Notes in Computer Science*, pages 38–42. Springer, 2010.
15. Chris Godsil and Gordon Royle. *Algebraic graph theory*, volume 207 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2001.
16. Frank Gurski. Characterizations for co-graphs defined by restricted nlc-width or clique-width operations. *Discrete Mathematics*, 306(2):271–277, 2006.
17. Frank Gurski. Linear layouts measuring neighbourhoods in graphs. *Discrete Mathematics*, 306(15):1637–1650, 2006.
18. Frank Gurski and Egon Wanke. On the relationship between nlc-width and linear nlc-width. *Theor. Comput. Sci.*, 347(1-2):76–89, 2005.
19. Frank Gurski and Egon Wanke. The nlc-width and clique-width for powers of graphs of bounded tree-width. *Discrete Applied Mathematics*, 157(4):583–595, 2009.
20. Pinar Heggernes, Daniel Meister, and Charis Papadopoulos. A complete characterisation of the linear clique-width of path powers. In Jianer Chen and S. Barry Cooper, editors, *TAMC*, volume 5532 of *Lecture Notes in Computer Science*, pages 241–250. Springer, 2009.
21. Pinar Heggernes, Daniel Meister, and Charis Papadopoulos. Graphs of linear clique-width at most 3. *Theor. Comput. Sci.*, 412(39):5466–5486, 2011.
22. Pinar Heggernes, Daniel Meister, and Charis Papadopoulos. Characterising the linear clique-width of a class of graphs by forbidden induced subgraphs. *Discrete Applied Mathematics*, 160(6):888–901, 2012.
23. Sang il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005.
24. Jisu Jeong, O.-joung Kwon, and Sang-il Oum. Excluded vertex-minors for graphs of linear rank-width at most k. *Eur. J. Comb.*, 41:242–257, 2014.
25. O joung Kwon. Connecting rank-width and tree-width via pivot-minors, 2012. Master's Thesis.
26. Lefteris M. Kirousis and Christos H. Papadimitriou. Searching and pebbling. *Theor. Comput. Sci.*, 47(3):205–218, 1986.
27. Ton Kloks and Hans L. Bodlaender. Approximating treewidth and pathwidth of some classes of perfect graphs. In Toshihide Ibaraki, Yasuyoshi Inagaki, Kazuo Iwama, Takao Nishizeki, and Masafumi Yamashita, editors, *ISAAC*, volume 650 of *Lecture Notes in Computer Science*, pages 116–125. Springer, 1992.
28. Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998.
29. Vadim V. Lozin and Dieter Rautenbach. The relative clique-width of a graph. *J. Comb. Theory, Ser. B*, 97(5):846–858, 2007.
30. Nimrod Megiddo, S. Louis Hakimi, M. R. Garey, David S. Johnson, and Christos H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, 1988.
31. Daniel Meister. Clique-width with an inactive label. *Discrete Mathematics*, 337:34–64, 2014.
32. Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms*, 5(1), 2008.
33. Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006.
34. Petra Scheffler. Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme. Akademie der Wissenschaften der DDR, Berlin, 1989. PhD thesis.
35. Karol Suchan and Ioan Todinca. Pathwidth of circular-arc graphs. In Andreas Brandstädt, Dieter Kratsch, and Haiko Müller, editors, *WG*, volume 4769 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2007.
36. Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Mathematics*, 127(1-3):293–304, 1994.