ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

From tree-decompositions to clique-width terms☆

Bruno Courcelle

Labri, CNRS and Bordeaux University, 33405 Talence, France

ARTICLE INFO

Article history:
Received 10 June 2016
Received in revised form 18 January 2017
Accepted 30 April 2017
Available online xxxx

Keywords:
Tree-width
Clique-width
Sparse graph
Planar graph
Incidence graph
Fixed-parameter tractable algorithm

ABSTRACT

Tree-width and clique-width are two important graph complexity measures that serve as parameters in many fixed-parameter tractable algorithms. We give two algorithms that transform tree-decompositions represented by normal trees into clique-width terms (a rooted tree is *normal* for a graph if its nodes are the vertices of the graph and every two adjacent vertices are on a path of the tree starting at the root). As a consequence, we obtain that, for certain classes of sparse graphs, clique-width is polynomially bounded in terms of tree-width. It is even linearly bounded for planar graphs and incidence graphs. These results are useful in the construction of model-checking algorithms for problems described by monadic second-order formulae, including those allowing edge set quantifications.

Introduction

Tree-width and clique-width are important graph complexity measures that occur as parameters in many *fixed-parameter tractable* (FPT) algorithms [16,18,20,21,22,25]. They are also important for the study of *graph structure* and, in particular, for the description of certain graph classes by forbidden subgraphs, minors or vertex-minors. Both notions are based on certain hierarchical graph decompositions, and the associated FPT algorithms use dynamic programming on these decompositions. Many of these algorithms need input graphs of moderate tree-width or clique-width that are given with the relevant decompositions. Constructing optimal decompositions is difficult [1,24], however, there exist polynomial time approximation algorithms [3,4,31].

A related problem consists in comparing width measures in the following way. Given two width measures wd and wd' and a graph class $\mathcal{C}$, we wish to prove that $wd'(G) \leq f(wd(G))$ for every graph G in $\mathcal{C}$, where f is a fixed function. We say that wd' is *linearly bounded* (resp. *polynomially bounded*) in terms of wd on $\mathcal{C}$, if f is linear (resp. polynomial). In view of algorithmic applications, it is also useful to have efficient algorithms to convert a decomposition witnessing $wd(G) \leq k$ into one witnessing $wd'(G) \leq f(k)$.

For the class of all graphs, clique-width[1] is not polynomially bounded in terms of tree-width [9], and tree-width is not bounded in terms of clique-width by any function. For several classes of sparse graphs (such graphs have $O(n)$ edges for n vertices, see Section 1), clique-width is polynomially bounded in terms of tree-width, and even linearly bounded for planar graphs, graphs of bounded degree and incidence graphs.[2] We will improve some known bounds, we will obtain bounds for

☆ This work has been supported by the French National Research Agency (ANR) within the IdEx Bordeaux program "Investments for the future", CPU, ANR-10-IDEX-03-02, and also within the project GraphEn started in October 2015. It will be presented, as part of an invited lecture, at the Workshop on graph classes, optimization, and width parameters (GROW 2017) organized by the Fields Institute in Toronto, Canada. I acknowledge support from this institute.

E-mail address: courcell@labri.fr.

1 Clique-width is defined algebraically from terms that define graphs. Such terms are called *clique-width terms*, see Definition 3. An alternative definition is in [17]. We denote the clique-width of a graph G by $cwd(G)$ and its tree-width by $twd(G)$.

2 The *incidence graph* $Inc(G)$ of a graph G is obtained from it by adding a new vertex on each edge.

http://dx.doi.org/10.1016/j.dam.2017.04.040
0166-218X/

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

directed graphs and we will give an algorithm that transforms tree-decompositions into clique-width terms. Together with combinatorial lemmas relative to each considered class, this algorithm yields the claimed linear or polynomial bounds. In the same framework, we will present the algorithm from [9] that gives a better exponential bounding for graphs of large clique-width.

Our algorithms will take as input tree-decompositions represented in a compact way by normal trees. A rooted tree is *normal* for a graph if its nodes are the vertices of the graph and adjacent vertices of the graph are comparable with respect to the ancestor relation of the tree. This representation is easier to implement (and perhaps also to visualize, but this is a subjective matter) than the pair (T, f) of the usual definition [2,16,19–21] recalled in Section 2. It works well for our algorithms and offers also an easy logical representation (as observed in [16], Example 5.2(4)). Normal trees fit with the contractions and deletions from which the quasi-order of minor inclusion is defined.

The inputs of our first algorithm will be, more generally, *quasi-normal trees*, defined similarly except that some nodes may not be vertices of the graph. The associated tree-decompositions will be obtained from the trees as by-products. We will give a definition of clique-width that relaxes constraints on the use of vertex labels and facilitates the construction of clique-width terms. These constructions have been implemented (see Section 6).

Many model-checking algorithms for tree-structured graphs (graphs of bounded tree-width, path-with, clique-width, etc.) use dynamic programming on terms or labelled trees that encode the relevant decompositions. In several articles [13,15], we have constructed FPT algorithms parameterized by clique-width for problems expressed in *monadic second-order logic* (MSO logic in short); these algorithms are based on *fly-automata*[3] taking clique-width terms as inputs. MSO formulae using *edge set quantifications* (called *MSO_2 formulae*) are more expressive than MSO formulae. However, the MSO_2 properties of a graph G are nothing but MSO properties of its incidence graph $Inc(G)$. As the clique-width of $Inc(G)$ is linearly bounded in terms of the tree-width of G, the algorithms for the MSO model-checking of graphs of bounded clique-width can be used (in practice) for the MSO_2 model-checking of graphs of bounded tree-width. This extension is developed in [11,12].

Summary and main results.

Section 1 reviews notation about trees and graphs. Section 2 defines our novel presentations of tree-decompositions and clique-width terms. Section 3 presents in a unified way three algorithms that convert tree-decompositions into clique-width terms. In Section 4, we obtain that $cwd(G) = O(twd(G))$ for planar graphs (we improve the linear bound given in [26]) and that $cwd(G) = O(twd(G)^q)$ for uniformly q-sparse graphs (the graphs whose subgraphs have at most $q \cdot n$ edges for n vertices). In Section 5, we consider q-hypergraphs (their hyperedges have at most q vertices). A q-hypergraph H can be viewed as a bipartite graph $Bip(H)$ and we prove that $cwd(Bip(H)) = O(twd(H)^{q-1})$. For incidence graphs, we deduce that $cwd(Inc(G)) = O(twd(G))$. In Section 6, we review the algorithmic applications to model-checking. In Appendix we consider the effect of minor-reducing operations on tree-decompositions defined by quasi-normal trees. We also compare our definitions with boolean-width [8].

1. Definitions and basic facts

Most definitions are well-known, we mainly review notation. We state a few facts that are either well-known or easy to prove.

The union of two disjoint sets is denoted by $\uplus$. The cardinality of a set X is denoted by $|X|$ and its powerset by $\mathcal{P}(X)$.

If $0 \leq m \leq k$, we define $\gamma(k, m)$ as the number of subsets of $[k] := \{1, \ldots, k\}$ of cardinality at most m. This number is $1 + k + \cdots + \binom{k}{m} = O(k^m)$ for fixed m. If $1 < m < k/2$, then $\gamma(k, m) < m\binom{k}{m} < k^m/(m-1)!$. We will actually use $\gamma(k, m)$ for "small", fixed values m and "large", variable ones k.

All trees, graphs and hypergraphs are nonempty and finite.

Trees

Trees are always rooted; N_T denotes the set of *nodes* of a tree T and $<_T$ its *ancestor relation*, a strict partial order on N_T (a node is not an ancestor of itself). The *root*, denoted by $root_T$, is the unique maximal element and the *leaves* are the minimal ones; $p_T(u)$ is the father (the closest ancestor) of a node u. We denote by $T_{\leq}(u)$ the set $\{w \in N_T \mid w \leq_T u\}$, by $T_<(u)$ the set $\{w \in N_T \mid w <_T u\}$ and similarly for $T_>(u)$ and $T_\geq(u)$. In particular, $T_>(u)$ is the set of ancestors of u. A tree is *binary* if every node has at most two sons.

If e is an edge of T between a node u and its father w, then the tree T' resulting from the *contraction of* e is constructed as follows: we remove u and e and we make each son of u into a son of w. The root of T' is that of T.

Graphs

Unless otherwise specified (e.g., in Section 5.2), we consider *simple graphs*, i.e., that are loop-free and without parallel edges. Graphs are directed or not. Undefined notions are as in [19]. A graph G has vertex set V_G and edge set E_G. If G is directed, E_G can be identified with the binary, irreflexive edge relation $edg_G \subseteq V_G \times V_G$; while being simple, G can have pairs of opposite edges. If G is undirected, then edg_G is symmetric and $|edg_G| = 2\,|E_G|$. The *undirected graph underlying* G is $Und(G)$ such that $V_{Und(G)} := V_G$ and $edg_{Und(G)} := edg_G \cup edg_G^{-1}$.

[3] The finite automata arising from MSO formulae are much too large to be implemented in the usual way by lists of transitions. In fly-automata, states and transitions are not tabulated but described by means of an appropriate syntax. Each time a transition is necessary, it is (re)computed. Only the transitions necessary for a given term are computed.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

We denote by $G[X]$ the induced subgraph of G with vertex set $X \cap V_G$, where X need not be a subset of V_G: this convention allows to deal easily with cases where X is a set of vertices of a graph H of which G is a subgraph.

If G is undirected and $x \in V_G$, then $N_G(x)$ denotes the set of *neighbours* of x; if G is directed, then $N_G^+(x)$ denotes the set of vertices y such that $x \to_G y$ (*i.e.*, there is an edge from x to y), $N_G^-(x)$, the set of those such that $y \to_G x$, and so, $N_G(x) := N_G^+(x) \cup N_G^-(x)$ is the set of neighbours of x. We extend these definitions to the case where $x \notin V_G$: then $N_G(x) := \emptyset$. For a set[4] X, $N_G^+(X)$ is the union of the sets $N_G^+(x)$, $x \in X$, and similarly for N_G^- and N_G.

If G is an undirected graph, and X, Y are disjoint sets, we define $\Omega(X, Y) := \{N_G(x) \cap Y \mid x \in X \cap V_G\}$. It is thus a set of subsets of $Y \cap V_G$. If G is directed, we define $\Omega(X, Y) := \{(N_G^+(x) \cap Y, N_G^-(x) \cap Y) \mid x \in X \cap V_G\}$. It is a set of pairs of subsets of $Y \cap V_G$. If G is undirected and Y has cardinality k, then $|\Omega(X, Y)| \leq 2^k$. If furthermore $1 \leq m \leq k$ and $|N_G(x)| \leq m$ for each $x \in X$, then $|\Omega(X, Y)| \leq \gamma(k, m) = O(k^m)$ for fixed m. If G is directed, then each edge of $Und(G)$ between x and y can come from three possible configurations of directed edges between these vertices. Hence, if $|Y| = k$, we have $|\Omega(X, Y)| \leq (1+3)^k = 2^{2k}$, and if $\left|N_{Und(G)}(x)\right| \leq m \leq k$ for each $x \in X$, then $|\Omega(X, Y)| < 3^m \gamma(k, m)$.

Sparse graphs

A graph G is *uniformly q-sparse* if $|E_H| \leq q\,|V_H|$ for every (undirected) subgraph H of $Und(G)$. An undirected graph G is uniformly q-sparse if and only if it has an orientation of indegree at most q ([27] or Proposition 9.40 of [16]). Every planar graph G is uniformly 3-sparse (because $|E_G| \leq 3\,|V_G| - 6$ if G is simple and planar). An undirected graph is uniformly $\lceil d/2 \rceil$-sparse if its maximum degree is d.

We denote by $\mathcal{S}_r$ the class of graphs G such that $Und(G)$ does not contain a subgraph isomorphic to the complete bipartite graph $K_{r,r}$. These graphs are $K_{r,r}$-free with respect to subgraph inclusion and removal of orientation. Every uniformly q-sparse graph belongs to $\mathcal{S}_{2q+1}$, but for every r and q, there are graphs in $\mathcal{S}_r$ that are not uniformly q-sparse (because there is a constant c such that, if $r \geq 3$, there is a graph having n vertices and at least $c \cdot n^{2-2/(r+1)}$ edges, see [19], Section 7.1).

2. Tree-width and clique-width

2.1. Tree-decompositions of various kinds

Tree-decompositions are well-known [2,19–21], but we review notation and we present new definitions concerning them. A *tree-decomposition* of a graph G is a pair (T, f) such that T is a rooted tree, f maps N_T to $\mathcal{P}(V_G)$, every vertex is in some *bag* $f(u)$, every edge has its two ends in some bag, and the *connectivity condition* holds: for every vertex x, the nodes u such that $x \in f(u)$ induce a connected subgraph of T. Its *width* $wd(T, f)$ is the maximum cardinality of a bag minus 1.

Our definitions are motivated by the following fact ([16], Proposition 2.67): every tree-decomposition can be transformed into one of same width such that the sets $f(root_T)$ and $f(u) - f(p_T(u))$ for all $u \in N_T - \{root_T\}$ are singleton. In such a case, one can identify N_T and V_G in such a way that $f(root_T) = \{root_T\}, f(u) - f(p_T(u)) = \{u\}$ and $u \leq_T w$ if $w \in f(u)$. We call normal such a tree-decomposition. The following definitions generalize this observation.

Definitions 1. Normal and quasi-normal, trees and tree-decompositions.

(a) A tree T is *quasi-normal for a graph G* if $V_G \subseteq N_T$ and the two ends of each edge of G are comparable under $<_T$. It is *normal* if, in addition, we have $V_G = N_T$. A depth-first spanning tree of a graph is thus normal.[5]

(b) A tree-decomposition (T, f) of a graph G is *quasi-normal* if $V_G \subseteq N_T$, $f(u) \subseteq T_{\geq}(u)$ for every $u \in N_T$, and furthermore, $u \in f(u)$ if $u \in N_T \cap V_G$. It is *normal* if, in addition, we have $V_G = N_T$. It is clear that T is quasi-normal (respectively normal) if (T, f) is so. We now prove a kind of converse.

(c) Let T be quasi-normal for a graph G. For each $u \in N_T$, we define the following sets:

$up_{G,T}(u) := N_G(u) \cap T_>(u)$ (this set is empty if $u \in N_T - V_G$),
$up^+_{G,T}(u) := N^+_G(u) \cap T_>(u)$ and $up^-_{G,T}(u) := N^-_G(u) \cap T_>(u)$ if G is directed,
$f^*_T(u) := N_G(T_{\leq}(u)) \cap T_>(u) = \bigcup\{up_{G,T}(w) \cap T_>(u) \mid w \leq_T u\}$,
$f_T(u) := \{u\} \cup f^*_T(u)$ if $u \in V_G$, and
$f_T(u) := f^*_T(u)$ if $u \in N_T - V_G$.

If $u \in V_G$, then $f_T(u)$ consists of u and its ancestors that are adjacent to some vertex $w \leq_T u$ and $f_T(u) = \varnothing$ if u is the root or a leaf, and is not in V_G.

We define the *width* of the pair (G, T) as the maximal cardinality of a set $f^*_T(u)$ for $u \in V_G$.

Claim 1. *If T is quasi-normal (resp. normal) for a graph G, then (T, f_T) is a quasi-normal (resp. normal) tree-decomposition of this graph. The width of the tree-decomposition (T, f_T) is that of (G, T).*

Proof. Every vertex u is in the bag $f_T(u)$. Each edge links some u to one of its ancestors, hence its two ends are in $f_T(u)$. The connectivity condition holds because, if $y \in f_T(u) \cap f_T(v)$, then $u, v \leq_T y$ and, by the definitions, y belongs to each set $f_T(w)$ such that $u \leq_T w \leq_T y$ or $v \leq_T w \leq_T y$, hence, it belongs to each set $f_T(w)$ for w on the undirected path between u and v.

[4] As for $G[X]$, the set X need not be a subset of V_G. The same holds for X, Y in $\Omega(X, Y)$ below.

[5] In [19], a normal tree for a graph must be a subgraph of this graph. We do not require this condition.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

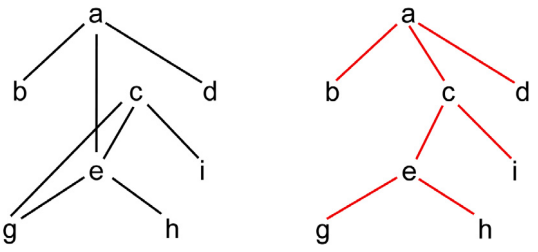

Fig. 1. A graph G and a normal tree T.

Let $u \in N_T - V_G$. If it is the root, then $f_T(u) = \varnothing$ and otherwise, $f_T(u) \subseteq f_T(p_T(u))$. It follows that the width of the tree-decomposition (T, f_T) is the maximal cardinality minus 1 of a set $f_T(u)$ for $u \in V_G$, hence is the width of (G, T). □

Claim 2. *If (T, f) is a quasi-normal tree decomposition, then $f_T(u) \subseteq f(u)$ for each u in N_T.*

Proof. Consider $y \in f_T(u)$. If $y = u$, then $y \in f(u)$. Otherwise, $x \leq_T u <_T y$ for some x adjacent to y, and $x, y \in f(w)$ for some node w. We have $w \leq_T x <_T y$, hence, $y \in f(u)$ by the connectivity condition since $w \leq_T u <_T y$ and $y \in f(y) \cap f(w)$. □

Informally, if T is quasi-normal, then f_T is the "minimal" mapping f such that (T, f) is a quasi-normal tree decomposition. In our constructions, we will mainly consider tree-decompositions (T, f_T) given by a normal or quasi-normal tree T.

Example 2. Fig. 1 shows a graph G and the tree T of a normal tree-decomposition (T, f). The function f is defined in the following table. The function f_T is equal to f except that $f_T(g) = \{c, e, g\} \subset f(g)$ and $f_T(h) = \{e, h\} \subset f(h)$. The set $f_T^*(c)$ contains vertex a because of the edge $a - e$. Clearly, (T, f) is not optimal, as (T, f_T) has smaller width (cf. Definitions 1(c)).

u	$f(u)$	u	$f(u)$
a	a	e	e, c, a
b	b, a	g	g, e, c, a
c	c, a	h	h, e, c
d	d, a	i	i, c

In further examples, we will use a linear notation for trees. The tree T of this example can be denoted by $a(b, c(e(g, h), i), d)$ and, equivalently, by $a(b, d, c(i, e(h, g)))$ as, in our trees, the sons of a node are not ordered. □

Lemma 3. *Every tree-decomposition can be transformed into a tree-decomposition of the same graph and of same width that is normal, and into one, similarly, that is quasi-normal and whose tree is binary.*

Proof sketch (*cf. [16], Proposition 2.67*)**.** If (T, f) is not normal, we first contract all edges $u - p_T(u)$ of T such that $f(u) \subseteq f(p_T(u))$ (cf. Section 1). Then, if $f(root_T) = \emptyset$, we contract the edge between $root_T$ and one of its sons, say w (we must have $f(w) \neq \emptyset$). We obtain a tree T_1 (its root is that of T) and we define f_1 as the restriction of f to N_{T_1}, except for the root if $f(root_T) = \emptyset$: in this case $f_1(root_T) := f(w)$ (where w is as above).

Then, for each node u such that $\left|f_1(u) - f_1(p_{T_1}(u))\right| = m > 1$, we insert $m - 1$ nodes forming a path with m edges between u and $p_{T_1}(u)$; similarly, if $|f_1(root_T)| = m > 1$, we insert $m - 1$ nodes below the root. We obtain a normal tree T_2 and tree-decomposition (T_2, f_2) of G of same width as (T_1, f_1) and (T, f). The function f_2 is easy to define and we identify a node u with the vertex x such that $f_2(u) - f_2(p_{T_2}(u)) = \{x\}$ or $f_2(u) = \{x\}$ if u is the root.

If T_2 is not binary, we transform it recursively into $\widetilde{T_2}$ as follows. If a node u of T_2 has sons $u_1, \ldots, u_p$ where $p \geq 3$, then we replace the subtree $u(U_1, \ldots, U_p)$ of T_2 by $u(\widetilde{U_1}, w_1(\widetilde{U_2}, w_2(\ldots, w_{p-2}(\widetilde{U_{p-1}}, \widetilde{U_p})\ldots)))$, where $w_1, w_2, \ldots, w_{p-2}$ are new nodes. We obtain a binary tree $T_3 := \widetilde{T_2}$. We have $f_{T_3}(u) = f_{T_2}(u)$ and $f_{T_3}(w_i) \subseteq f_{T_2}(u)$ for $u, w_1, w_2, \ldots, w_{p-2}$ as above (u is a vertex of G). Hence (T_3, f_{T_3}) is quasi-normal with a binary tree, as wanted. Its width is that (T_2, f_2) and (T, f). □

If in this proof (T, f_T) is quasi-normal but not normal, then the transformation contracts all edges $u - p_T(u)$ of T such that $u \in N_T - V_G$ and possibly one edge incident to the root. The obtained tree T_1 is normal for G and (T_1, f_1) is a normal tree-decomposition.

For using optimal tree-decompositions, there is no loss of generality in considering only normal ones with "minimal" bags, hence tree-decompositions of the form (T, f_T) where T is normal for G. Such a tree-decomposition can be encoded in a very compact way: the tree T is represented by the partial function $p_T : V_G \to V_G$ (defined by pointers or any other appropriate data structure), the edges of G by the function $up_{G,T}$ (or by $up^+_{G,T}$ and $up^-_{G,T}$) and the sets $f_T(u)$ for $u \in N_T = V_G$ can

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

be computed from edg_G, from $up_{G,T}$ or from $up^+_{G,T}$ and $up^-_{G,T}$, in time $O(m)$ where m is the size of (T, f_T) defined as the sum of cardinalities of the sets $f_T(u)$ for $u \in N_T$. Clearly, $m \leq (k+1)\,|V_G|$ where k is the width of (T, f_T).

Definition 4. Clean tree-decompositions.

A normal tree-decomposition (T, f) of a graph G is *clean* if $f = f_T$ and $p_T(u) \in f(u)$ for every node u of T that is not the root.

This is the case if T is a depth-first spanning tree and $f = f_T$. In Example 2, the decomposition (T, f_T) is clean but T is not spanning (the edge $a - c$ is not in G whereas $a \in f_T(c)$. For another example, consider $K_{1,2}$ with vertex 1 adjacent to vertices 2 and 3 and normal tree $U = 1 \to 2 \to 3$ with root 1. The tree-decomposition (U, f_U) is normal (and optimal) but not clean because the father of 3 is 2 and $f_U(3) = \{1, 3\}$.

Claim. A graph is connected if it has a clean tree-decomposition.

Proof sketch. By using bottom-up induction on u in N_T, one proves that each graph $G[T_{\leq}(u)]$ is connected. This fact holds because $p_T(w) \in f^*_T(w)$ for each $w \in N_T = V_G$, hence u is linked to $G[T_{\leq}(w)]$ if $u = p_T(w)$, so that $G[T_{\leq}(u)]$ is connected. □

Lemma 5. *From every normal tree-decomposition (T, f) of a connected graph G, one can construct in time $O(|E_G| \,.\, |N_T|)$ a clean tree-decomposition of G of no larger width.*

Proof. Let (T, f) be a normal tree-decomposition of a connected graph G. We first compute f_T in time $O(|E_G| \,.\, |N_T|)$. Since G is connected, no set $f^*_T(u)$ is empty, except if u is the root (because then $f_T(u) = \{u\}$). For each $u \in N_T$ such that $p_T(u) \notin f_T(u)$, we let $\widetilde{u}$ be the least element of $f^*_T(u)$ with respect to $\leq_T$. We modify T by making $\widetilde{u}$ the father of u, and we let T' be the new tree. Then (T', f_T) is a clean tree-decomposition of G of same width as (T, f_T), which is no larger than that of (T, f).

If each set $f^*_T(u)$ is implemented as a list in increasing order with respect to $\leq_T$, then $\widetilde{u}$ is accessed in constant time, and so, we can construct T' in time $O(|N_T|)$. □

Hence, every connected graph has an optimal tree-decomposition that is clean. Clean tree-decompositions (used in [9] and below in Algorithm 12) arise in a natural way from the notion of *partial k-tree* that we now recall.

An undirected graph G is *chordal* if it is connected, its vertices can be denoted by $1, \ldots, n$ in such a way that $G[N_G(i) \cap [i-1]]$ is a clique[6] for each $i = 2, \ldots, n$ (this is one definition among others, cf. [16], Proposition 2.72). A normal tree-decomposition (T, f_T) is obtained as follows: T has nodes $1, \ldots, n$, $root_T := 1$ and $p_T(i) := \min(N_G(i) \cap [i-1])$. We have $f^*_T(i) = N_G(i) \cap [i-1]$. This tree-decomposition is optimal and clean.

A partial k-tree is a graph obtained by edge deletions from a chordal graph G of maximal clique size $k+1$. A graph H has tree-width at most k if and only if $Und(H)$ is a partial k-tree. The tree-decomposition (T, f_T) of G is a normal tree-decomposition of H. This tree-decomposition may not be clean, but we can clean it by Lemma 5.

Definition 6. Special tree-decompositions.

A tree-decomposition (T, f) is *special* if any two nodes u, u' of T such that $f(u) \cap f(u') \neq \emptyset$ are comparable with respect to $\leq_T$, equivalently, if $f(u) \cap f(u') = \emptyset$ for any two distinct nodes u, u' with same father. Equivalently, the nodes u such that $f(u)$ contains a given vertex form a directed path in the rooted tree T. We get the notion of *special tree-width*, denoted by $sptwd$. This notion has been introduced in [10]. It is clear that $twd(G) \leq sptwd(G)$ but graphs of tree-width 2 have unbounded special tree-width. Graphs of special tree-width 2 have been studied in [5–7].

Every special tree-decomposition can be made normal and special without increasing its width by the algorithm of Lemma 3. However, it cannot always be made clean and special: the star $K_{1,3}$ has a special tree-decomposition (actually a path-decomposition) of width 1 but no clean and special tree-decomposition of this width. We will give a simple proof that $cwd(G) \leq sptwd(G) + 2$ for every graph G, a result from [10].

In Appendix, we will examine how quasi-normal tree-decompositions behave with respect to the minor quasi-order. We will not need the corresponding facts for our main results, but they prove that the notions of normal and quasi-normal tree-decompositions fit well with the theory of tree-width.

2.2. Clique-width

Clique-width is a graph complexity measure defined from operations that construct graphs equipped with vertex labels. We review definition and notation, and we establish a technical result.

Definition 7. Clique-width

[6] We recall that $[i-1]$ denotes the set $\{1, \ldots, i-1\}$. Integers are here vertices.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

(a) Let C be a finite or infinite set of labels. A *C-graph* is a triple $G = (V_G, edg_G, \pi_G)$ where π_G is a mapping: $V_G \to C$. Its *type* is $\pi(G) := \pi_G(V_G)$, *i.e.*, the finite set of labels from C that label some vertex of G.

We denote by $\simeq$ the isomorphism of C-graphs up to vertex labels, *i.e.*, the isomorphism of the underlying unlabelled graphs.

A mapping $h : C \to C$ is *finite* if $h(a) \neq a$ for finitely many labels a. It can be specified in a finitary way by listing the pairs $(a, h(a))$ such that $h(a) \neq a$. We denote by $\Delta(h)$ the set of these pairs.

We define F_C as the following set of operations on C-graphs:

the union of two disjoint C-graphs, denoted by the binary function symbol $\oplus$,

the unary operation $relab_h$ that changes every vertex label a into $h(a)$ where h is a finite mapping from C to C,

the unary operation $\overrightarrow{add}_{a,b}$, for $a, b \in C, a \neq b$ that adds an edge from each a-labelled vertex x to each b-labelled vertex y (unless there is already an edge $x \to y$)

and, for each $a \in C$, the nullary symbol $\mathbf{a}(x)$ that denotes the isolated vertex x labelled by a.

For building undirected graphs, we use similarly $add_{a,b}$ to add undirected edges. In a well-formed term t, no two occurrences of nullary symbols denote the same vertex.[7]

Every term t in $T(F_C)$ is called a *clique-width term*. It denotes a C-graph $G(t)$. We will denote $\pi(G(t))$ by $\pi(t)$ and call it the *type* of t. The equivalence on terms $t \simeq t'$ is defined as $G(t) \simeq G(t')$ and $t \equiv t'$ as $G(t) = G(t')$ (vertices are specified in the terms t, t').

The *clique-width* of a graph G, denoted by $cwd(G)$, is the least cardinality of a set C such that $G \simeq G(t)$ for some $t \in T(F_C)$. It is frequently convenient to take $C = [k]$.

As $\oplus$ is associative, we will use it as an operation of variable arity. For readability, we will write $t = t_1 \oplus t_2 \oplus \ldots \oplus t_n$ instead of $\oplus(t_1, t_2, \ldots, t_n)$, defined as a shorthand for $t_1 \oplus (t_2 \oplus (\ldots \oplus t_n)\ldots)$. We define the size $|t|$ of t as $|t_1| + \cdots + |t_n| + n - 1$. If h only changes a into b, we denote $relab_h$ by $relab_{a\to b}$ and call this operation an *elementary relabelling*. By using only elementary relabellings, we obtain the same notion of clique-width ([16], Proposition 2.118). A relabelling $relab_h$ is *bijective on a term t* if h is injective on $\pi(t)$, hence is a bijection: $\pi(t) \to h(\pi(t))$. (See Section 6 about the use of these notions.)

(b) Our proofs will use a characterization[8] of clique-width allowing easy constructions of clique-width terms. If $u \in Pos(t)$, *i.e.*, is a position in a term $t \in T(F_C)$, then the subterm of t issued from u, denoted by t/u, denotes a C-graph $G(t/u)$ that is, up to vertex labels, a subgraph of $G(t)$ (because we use nullary symbols $\mathbf{a}(x)$ to designate vertices). Hence, $G(t) = G(t/root_T)$. We define the *width of* t as $wd(t) := \max\{|\pi(t/u)| \mid u \in Pos(t)\} \leq |C|$.

If k labels occur in a term t, then $G(t)$ has clique-width at most k. However, k can be an overestimation of $cwd(G(t))$. The upper-bound to $cwd(G(t))$ that arises from t is actually $wd(t)$ defined as the maximum number of labels that occur in a graph $G(t/u)$ for any position u in t. This is proved in the next proposition.

Proposition 8. *The clique-width of a graph is the minimal width of a term that defines it, up to vertex labels and isomorphism. Every clique-width term t can be transformed into an $\simeq$-equivalent term t' in $T(F_{[wd(t)]})$.*

Proof. Let $t \in T(F_C)$, $G = G(t)$ and $k = wd(t)$.

First step. By replacing in t each subterm $\oplus(t_1, t_2, \ldots, t_n)$ by $t_1 \oplus (t_2 \oplus (\cdots \oplus t_n) \cdots)$, we get a $\equiv$-equivalent term of same size as t where all occurrences of $\oplus$ are binary.

Second step. We fix t obtained by the first step and we denote $\pi(t/u)$ by $\pi(u)$. We now compute in a bottom-up way the following items, for each $u \in Pos(t)$:

the set $\pi(u)$,

the number $k_u := \max\{|\pi(w)| \mid w \leq u\} \leq k$,

an injective mapping $h_u : \pi(u) \to [k_u]$ such that $t_u \equiv relab_{h_u}(t/u)$.

The desired term t' will be t_r where r is the first position of t (the root of its syntactic tree). The bottom-up computation uses the following clauses:

(1) If u is an occurrence of $\mathbf{a}(x)$, then $\pi(u) = \{a\}$, $k_u = 1$ and h_u maps a to 1.

(2) If $t/u = \overrightarrow{add}_{a,b}(t/w)$, then $\pi(u) = \pi(w)$, $k_u = k_w$ and we define $h_u := h_w$. If a or b is not in $\pi(w)$, then the operation $\overrightarrow{add}_{a,b}$ has no effect and we define $t_u := t_w$. Otherwise, we define $t_u := \overrightarrow{add}_{h_w(a),h_w(b)}(t_w)$.

(3) If $t/u = relab_h(t/w)$, then $\pi(u) = h(\pi(w))$, $k_u \leq k_w$; we take for h_u any[9] injective mapping: $\pi(u) \to [k_u]$ and we define $t_u := relab_{h'}(t_w)$ where $h' := h_u \circ h \circ h_w^{-1}$.

(4) If $t/u = t/w \oplus t/w'$, then $\pi(u) = \pi(w) \cup \pi(w')$, $k_u = |\pi(u)| \geq k_w, k_{w'}$ and we take for h_u any injective mapping: $\pi(u) \to [k_u]$ whose restriction to $\pi(w)$ is h_w and we define $t_u := t_w \oplus relab_h(t_{w'})$ where $h := h_u \circ h_{w'}^{-1}$.

The verification that $t_u \equiv relab_{h_u}(t/u)$ is straightforward by the same induction. □

This proposition simplifies the construction of clique-width terms because, in a first step, we can use infinite sets C of labels, and then, in a second step, we can use it to transform an obtained term in $T(F_C)$, say t, into an $\simeq$-equivalent term in

[7] One can also use nullary symbols $\mathbf{a}$ that do not designate any particular vertex. In that case, the vertex defined by an occurrence u of $\mathbf{a}$ in the term is u itself. See [16], Section 2.52.

[8] It is used implicitly in [9], however, we think useful to detail it. See also Section 6 for its use in fly-automata.

[9] The mapping h_u can be chosen so that the corresponding mapping h' changes as few labels as possible.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

$T(F_{\lfloor wd(t)\rfloor})$. However, we will see in Section 6 that *fly-automata*, as defined in [13] can use terms of "small" width belonging to $T(F_C)$ where C is "large".

One can also construct the terms t_u in such a way that $\pi(t_u) = [|\pi(t_u)|]$. To do that we choose bijections $h_u : \pi(u) \to [|\pi(t_u)|]$ in Clauses (3) and (4) (we recall that $[|\pi(t_u)|] \subseteq [k_u]$). This strengthening is not crucial for using fly-automata.

2.3. Comparisons between tree-width and clique-width

For every directed graph G, we have $twd(Und(G)) = twd(G)$ and $cwd(Und(G)) \leq cwd(G)$ ([16], Proposition 2.105(3)).

Theorem 9. *For all graphs G, the following hold:*
(1) *$twd(G)$ is unbounded in terms of $cwd(G)$,*
(2) *$cwd(G) \leq 3 \cdot 2^{twd(G)-1}$ if G is undirected,*
(3) *$cwd(G) \leq 2^{2twd(G)+2} + 1$ if G is directed.*
(4) *There is no constant a such that $cwd(G) = O(twd(G)^a)$ for all graphs G.*

Proof. Cliques have clique-width 2 and unbounded tree-width, which proves (1). Assertions (2) and (3) are proved respectively in [9] and in Proposition 2.114 of [16]. For each k, there exists an undirected graph of tree-width $2k$ and clique-width larger than 2^{k-1} (a result from [9]). This proves Assertion (4). □

We will give a construction that proves easily (2) and improves the bound of (3).

Theorem 10. (1) *If G has maximal degree at most d (with $d \geq 1$), we have:*
(1.1) $twd(G) \leq 3d \cdot cwd(G) - 1$,
(1.2) $cwd(G) \leq 20d(twd(G) + 1) + 2$.
(2) *If $r \geq 2$ and $Und(G)$ has no subgraph isomorphic to $K_{r,r}$, then $twd(G) \leq 3(r-1)cwd(G) - 1$.*

Proof. Assertion (2) is from [28] (also [16], Proposition 2.115) and it yields (1.1). Assertion (1.2) is proved in [10] by means of a strong result of [32]. □

Unlike the bounds of Theorem 9(2,3), that of Theorem 10(1.2) is the same for directed and undirected graphs. Theorem 10(2) shows that, for each q, $twd(G) = O(cwd(G))$ if G is uniformly q-sparse. An opposite (polynomial) bounding will be established in Theorem 19.

3. From tree-decompositions to clique-width terms

Most FPT algorithms parameterized by tree-width or clique-width take as input a tree-decomposition of the considered graph or a clique-width term defining it. Unfortunately, tree-width and clique-width (and the corresponding optimal decompositions and terms) are difficult to compute[10] [1,24], but there exist polynomial time approximation algorithms.

There is a rich literature on efficient algorithms that construct tree-decompositions [4], but not so for clique-width.[11] For many graphs, *e.g.* rectangular grids, clique-width and tree-width are equal, up to a small fixed constant. It is thus useful to transform tree-decompositions[12] into clique-width terms. Theorem 9 is discouraging on first sight because of the exponential boundings, but for a number of useful graph classes (not only for graphs of bounded degree, cf. Theorem 10), we have $cwd(G) = O(twd(G)^q)$, with even $q = 1$ for planar graphs and incidence graphs. We will give two algorithms that work for all types of graphs, and we will obtain such bounds from the first of them. The second one is more interesting for certain graphs of large clique-width.

Theorem 11. *Let (T, f) be a quasi-normal tree-decomposition of a graph G. If $|\Omega(T_<(u), f(u))| \leq m$ for every node u of T, then $cwd(G) \leq m + 1$. A clique-width term witnessing this bound can be constructed from (T, f) in linear time.*

The value $|\Omega(T_<(u), f(u))|$ is somewhat related with boolean-width [8]. We will discuss this point in Appendix.

Before starting the proof, we explain its idea. We proceed bottom-up in the tree T. For each node u, we define a graph $H(u)$ consisting of $G[T_\leq(u)]$, each vertex w of which has label $N_G(w) \cap T_>(u)$. This label indicates to which ancestors of u the vertex w will have to be linked in further stages of the construction. From the definitions, the vertices of $H(u)$ are linked in G, outside of $H(u)$, to vertices in $f_T^*(u)$, a subset of $T_>(u)$. Labels are useful to construct $H(u)$ from $H(u_1), \ldots, H(u_p)$ where $u_1, \ldots, u_p$ are the sons of u. Actually, we will construct for each u a clique-width term denoting $H(u)$. Its labels are sets of vertices, hence the set of labels has unbounded cardinality. But this is allowed by Proposition 8, and we avoid to obscure the construction with relabellings. The maximum number of labels occurring in $H(u)$ is the number of sets of the form $N_G(w) \cap T_>(u) = N_G(w) \cap f_T^*(u)$ for $w \leq_T u$. We now detail the proof.

[10] It is possible to decide in linear time if a graph G of tree-width k has clique-width at most m, for fixed k and m [23], but the complicated algorithm does not highlight the structural properties of G ensuring that $cwd(G) \leq m$.

[11] A good linear-time approximation algorithm for tree-width is in [3]. The cubic-time approximation algorithm of [31] produces a clique-width term of width at most 8^k for given k and G of clique-width at most k.

[12] By Definitions 1, a quasi-normal tree can be taken as input of an algorithm doing that.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

Proof. We are given a quasi-normal tree-decomposition (T, f) of a graph G, hence T is quasi-normal and $V_G \subseteq N_T$. We define $f^*(u) := f(u) - \{u\}$ for $u \in N_T$.

We first assume that G is undirected. We will construct a term $t \in T(F_C)$ that defines G by using the set of labels $C := \{*\} \uplus \mathcal{P}(V_G)$. For each $u \in N_T$, we define $H(u)$ as the graph $G[T_{\leq}(u)]$ where each vertex w has label $N_G(w) \cap T_>(u)$ (in particular, u has label $up_{G,T}(u)$). We have $N_G(w) \cap T_>(u) \subseteq f_T^*(u) \subseteq f^*(u)$ and $\pi(H(u)) = \Omega(T_{\leq}(u), f^*(u))$. In particular, $H(root_T)$ is G with all vertices labelled by $\emptyset$.

The following inductive characterization of $H(u)$ yields immediately a term t_u in $T(F_C)$ that denotes it.

If u is a leaf, then $H(u) = \mathbf{c}(u)$, where $c := up_{G,T}(u) = N_G(u)$.

Otherwise, u has sons $u_1, \ldots, u_p$; then, if $u \notin V_G$ we have:

$H(u) = H(u_1) \oplus \ldots \oplus H(u_p)$. (The label of a vertex w in $H(u_i)$ is $N_G(w) \cap T_>(u_i) = N_G(w) \cap T_>(u)$, hence is the same in $H(u)$.)

If $u \in V_G$ we have:

$H(u) = relab_{*\to c}(relab_h(A(*(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p))))$ where

$c := up_{G,T}(u)$, A is the composition of the operations $add_{*,d}$ such that $u \in d$ and $d \in \pi(H(u_1) \oplus \ldots \oplus H(u_p))$, and $h(d) := d - \{u\}$ for all sets d as above ($h(d) := d$ for all other $d \in C$).

The correctness is clear from the definition. We now bound the width of the terms t_u. For doing that, we need to bound the cardinalities of the types of their subterms.

If u is a leaf, then $|\pi(H(u))| = |\pi(t_u)| = 1$ because $t_u = \mathbf{c}(u)$.

Otherwise, by the definitions, $\pi(H(u)) = \pi(t_u)$ is the set of sets $N_G(w) \cap T_>(u)$ for $w \leq_T u$. As already noted, $\pi(H(u)) = \Omega(T_{\leq}(u), f^*(u))$.

We first consider the case where $u \in V_G$. We have:

$$\pi(H(u_i)) = \Omega(T_{\leq}(u_i), f^*(u_i)) = \Omega(T_{\leq}(u_i), f(u)) \subseteq \Omega(T_<(u), f(u)). \tag{1}$$

Hence:

$$\pi(A(*(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p))) = \pi(*(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p)) \subseteq \{*\} \cup \Omega(T_<(u), f(u)), \tag{2}$$

$$\pi(relab_h(A(*(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p)))) \subseteq \{*\} \cup \Omega(T_<(u), f^*(u)). \tag{3}$$

We have also:[13]

$$\pi(H(u)) = \{c\} \cup \Omega(T_<(u), f^*(u)) = \Omega(T_{\leq}(u), f^*(u)). \tag{4}$$

If $u \notin V_G$, the situation is simpler because $\pi(H(u)) = \pi(H(u_1) \oplus \ldots \oplus H(u_p))$ and

$$\pi(H(u)) = \Omega(T_<(u), f(u)) = \Omega(T_{\leq}(u), f(u)). \tag{5}$$

We now bound the cardinalities of these sets of labels. We have:

$$\left|\Omega(T_<(u), f^*(u))\right| \leq |\Omega(T_<(u), f(u))| \leq m, \quad \text{hence } |\pi(H(u))| \leq m + 1$$

for each u, by (4) and (5). We also have:

$$\left|\{*\} \cup \Omega(T_<(u), f^*(u))\right| \leq 1 + |\Omega(T_<(u), f(u))| \leq m + 1. \tag{6}$$

Since, by induction, $wd(t_{u_i}) \leq m + 1$ for each i, we have $wd(t_u) \leq m + 1$.

We now consider the case where G is directed. The proof is similar with $C := \{*\} \uplus (\mathcal{P}(V_G) \times \mathcal{P}(V_G))$. For each $u \in N_T$, we define $H(u)$ as the graph $G[T_{\leq}(u)]$ where each vertex w has label $(N_G^+(w) \cap T_>(u), N_G^-(w) \cap T_>(u))$. Note that $(N_G^+(w) \cup N_G^-(w)) \cap T_>(u) \subseteq f^*(u)$. The inductive characterization of $H(u)$ is as follows:

If u is a leaf, then $H(u) = \mathbf{c}(u)$, where $c := (up_{G,T}^+(u), up_{G,T}^-(u))$.

Otherwise, u has sons $u_1, \ldots, u_p$. Then, if $u \notin V_G$ we have:

$H(u) = H(u_1) \oplus \ldots \oplus H(u_p)$ and, if $u \in V_G$:

$H(u) = relab_{*\to c}(relab_h(A(*(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p))))$ where

$c := (up_{G,T}^+(u), up_{G,T}^-(u))$, A is the composition of the operations $\overrightarrow{add}_{(d,d'),*}$ such that $(d, d') \in \pi(H(u_1) \oplus \ldots \oplus H(u_p))$ and $u \in d$, and of the operations $\overrightarrow{add}_{*,(d,d')}$ such that $(d, d') \in \pi(H(u_1) \oplus \ldots \oplus H(u_p))$ and $u \in d'$, and, finally, $h(d, d') := (d - \{u\}, d' - \{u\})$ for all pairs (d, d') as above.

The correctness is clear from the definition. We now bound the width of the terms t_u. By the definitions, $\pi(H(u)) = \pi(t_u)$ is the set of pairs $(N_G^+(w) \cap T_>(u), N_G^-(w) \cap T_>(u))$ for $w \leq_T u$ and we also have $(N_G^+(w) \cup N_G^-(w)) \cap T_>(u) \subseteq f(u) - \{u\}$ for each such w. Hence $\pi(H(u)) = \Omega(T_{\leq}(u), f^*(u))$.

If u is a leaf, then $|\pi(H(u))| = |\pi(t_u)| = 1$ as $t_u = \mathbf{c}(u)$.

[13] The set c is empty if all edges of G incident to u are in $H(u)$. Actually, we know Equality (4) already from the definitions.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

Otherwise u has sons $u_1, \ldots, u_p$ and we have, similarly as (1):

$$\pi(H(u_i)) = \Omega(T_{\leq}(u_i), f^*(u_i)) = \Omega(T_{\leq}(u_i), f(u)) \subseteq \Omega(T_{<}(u), f(u)).$$

Inequalities and equalities (2)–(6) hold in the same way, and we have $wd(t_u) \leq m+1$. □

Remarks. (1) We have $\Omega(T_{\leq}(u), f_T^*(u)) = \Omega(T_{\leq}(u), f^*(u))$ for all u. It follows that the construction is exactly the same if we replace f by f_T; we still have $|\Omega(T_{<}(u), f_T(u))| \leq m$ for every node u. By Lemma 3, we can also transform (T, f) into a normal tree-decomposition.

(2) Let us examine the description of $H(u)$ for $u \in V_G$. If $c := up_{G,T}(u) \notin \pi(H(u_1) \oplus \ldots \oplus H(u_p))$, we have the simpler expression:

$H(u) = relab_h(A(\mathbf{c}(u) \oplus H(u_1) \oplus \ldots \oplus H(u_p)))$ where
A is the composition of the operations $add_{c,d}$ such that
$d \in \pi(H(u_1) \oplus \ldots \oplus H(u_p))$ and $u \in d$,
$h(d) := d - \{u\}$ for all sets d as above.

(3) Assume that $|\Omega(T_{<}(u), f(u))| \leq m$ for every node u of T that is not binary. Let (T', f') is a binary quasi-normal tree-decomposition constructed by the method of Lemma 3. Then $\left|\Omega(T'_{<}(u), f'(u))\right| \leq m$ for every node u of T'.

(4) The construction of Theorem 11 does not use the full power of $add_{a,b}$ and $\overrightarrow{add}_{a,b}$ because these operations are applied (in the terms t_u) to graphs having only one vertex labelled by a or only one vertex labelled by b. Hence, we do not obtain optimal clique-width terms. For the graphs $K_{n,n}$ of clique-width 2, given by clean optimal tree-decompositions, we obtain clique-width terms of width 3. (Remark 2 does not apply.) □

As an immediate consequence, we get that if G is undirected of tree-width k, then $cwd(G) \leq 2^{k+1}+1$ because then $|f(u)| \leq k+1$, so that $|\Omega(T_{<}(u), f(u))| \leq 2^{k+1}$ for each u; if G is directed, then $cwd(G) \leq 2^{2k+2}+1$. (If G is undirected, we have actually $cwd(G) \leq 2^{k+1}$ because, if $|f(u)| = k+1$, then $f(u) \notin \Omega(T_{<}(u), f(u))$. The second example below shows that the upper-bound 2^{k+1} on the width of the constructed term can be reached.) However, we will obtain better bounds by means of Algorithm 12.

Examples: (1) If we apply this construction to the clique K_n of tree-width $n-1$, by using Remark 2 above, we get an optimal clique-width term of width 2.

(2) Let G be the undirected graph with vertex set $[k+1] \uplus P$ where $P := \mathcal{P}([k+1]) - [k+1]$. Its edges are $i-j$ for $1 \leq i < j \leq k+1$, $d-i$ for all $d \in P$, $i \in d$, and $\emptyset - \{1\}$. Let T be the normal tree[14] $1(2(\ldots(k+1(d_1(\emptyset), d_2, \ldots, d_p)\ldots)))$ where $\{d_1, d_2, \ldots, d_p\} = P - \{\emptyset\}$ and $d_1 = \{1\}$. Then (T, f_T) is a normal tree-decomposition of width k. (It is optimal because the set of vertices $[k+1]$ induces a clique.) For $u := k+1$, we have $\Omega(T_{<}(u), f_T(u)) = P$ and the other sets $\Omega(T_{<}(u), f_T(u))$ are smaller. Hence, the constructed term has width $|P|+1 = 2^{k+1}$. However, it is not hard to construct a clique-width term for G of width $k+3$. Hence, an optimal tree-decomposition does not produce necessarily an optimal clique-width term.

(3) The following example shows that different optimal tree-decompositions can yield clique-width terms of different width.

We let H be the undirected graph with vertex set $[k] \uplus Q$ where $Q := \{a_1, \ldots, a_k\}$. Its edges are $i-j$ for $1 \leq i < j \leq k$, $a_i - j$ for all $1 \leq j < i$. Let T be the normal tree $1(2(\ldots(k(a_1, a_2, \ldots, a_k)\ldots)))$. Then (T, f_T) is a normal tree-decomposition of width k that is optimal (because of the $(k+1)$-clique induced by $1, \ldots, k, a_k$) but not clean. For $u := k$, we have $\Omega(T_{<}(u), f_T(u)) = \{[1], \ldots, [k]\}$ (note that $[1], \ldots, [k]$ are sets of vertices) and the other sets $\Omega(T_{<}(u), f_T(u))$ are smaller. Hence, the constructed term has width $k+1$.

The corresponding clean decomposition has tree $T' = 1(a_1, 2(a_2, \ldots(k(a_k)\ldots)))$. We have $\Omega(T'_{<}(i), f_{T'}(i)) = \{[i]\}$ for each i, hence, we obtain a clique-width term of width 2, which is optimal.

That the tree-decomposition is clean is not enough to ensure that the constructed term has small width. Consider the tree $T'' = k(\ldots(2(1(a_1, a_2, \ldots, a_k)\ldots)))$. The corresponding optimal tree-decomposition is clean but $\Omega(T''_{<}(k), f_{T''}(k)) = \{[1], \ldots, [k]\}$ and $\Omega(T''_{<}(i), f_{T''}(i)) = \{\emptyset, [1], \ldots, [i]\}$ for each $i = 1, \ldots, k-1$. We obtain a term of width $k+1$. □

Algorithm 12. Another construction of clique-width terms from tree-decompositions.

The input is a normal tree-decomposition (T, f) of a graph G such that $|\Omega(T_{<}(u), f(u))| \leq m$ and $|\Omega(T_{<}(u), f^*(u))| \leq m'$ for each $u \in N_T$. The output is a clique-width term denoting G, whose width is at most $m+m'+1$.

[14] The linear notation of trees is described in Example 2.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

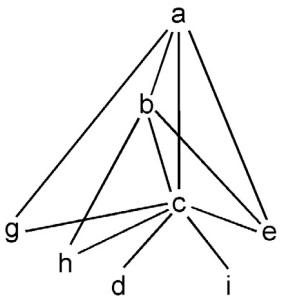

Fig. 2. A graph of clique-width 2 and tree-width 3.

Method.

We first consider G undirected. As in Theorem 11, we construct a term $t \in T(F_C)$ that denotes G where $C := \{*\} \uplus \mathcal{P}(V_G)$. We use the same graphs $H(u)$ but we construct them inductively in a different way.

If u is a leaf, then $H(u) = \mathbf{c}(u)$, where $c := up_{G,T}(u) = N_G(u)$.

Otherwise, u (it is in V_G) has sons $u_1, \ldots, u_p$. Then, we define:

$L_1 := relab_{h_1}(A_1(H(u_1) \oplus *(u)))$,

$L_i := relab_{h_i}(A_i(H(u_i) \oplus L_{i-1}))$, for $i = 2, \ldots, p$,

$H(u) = relab_{* \to c}(L_p)$,

where:

A_i is the composition of the operations $add_{*,d}$ such that $u \in d$ and $d \in \pi(H(u_i))$,

$h_i(d) := d - \{u\}$ for all d as in the definition of A_i.

Correctness is clear. This characterization yields a term t_u that denotes $H(u)$. We now bound the width of these terms t_u, and for that, we examine the types of their subterms.

We have $\pi(H(u)) = \Omega(T_{\leq}(u), f^*(u))$ for each u. In the above characterization:

$\pi(L_i) = B_i \cup \{*\}$ for all $i = 1, \ldots, p$, where

$B_i := \Omega(T_{\leq}(u_1), f^*(u_1) - \{u\}) \cup \ldots \cup \Omega(T_{\leq}(u_i), f^*(u_i) - \{u\})$,

$\pi(H(u_i) \oplus L_{i-1}) = \Omega(T_{\leq}(u_i), f^*(u_i)) \cup B_{i-1} \cup \{*\}$.

The largest of these sets are those of the form $\pi(H(u_i) \oplus L_{i-1})$. We have $\Omega(T_{\leq}(u_i), f^*(u_i)) \subseteq \Omega(T_{<}(u), f(u))$ and $B_{i-1} \subseteq \Omega(T_{<}(u), f^*(u))$, which gives $|\pi(H(u_i) \oplus L_{i-1})| \leq m + m' + 1$.

Hence, the terms t_u have width at most $m + m' + 1$.

For directed graphs, the proof is the same by modifying the operations A_i, similarly as we did in the second part of the proof of Theorem 11. □

Example. Let G be the graph of tree-width 3 of Fig. 2 and let $T = a(b(c(d, e, g, h, i)))$. Then (T, f_T) is a clean and optimal tree-decomposition of G. The construction of Theorem 11 yields a clique-width term of width 5 because $\Omega(T_{<}(c), f_T(c)) = \{\{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$ and the other similar sets are strictly smaller.

Let us now use Algorithm 12. For $u := c$, we let $(u_1, \ldots, u_5) := (d, e, g, h, i)$. Then $\pi(H(u_5) \oplus L_4) = \{\{c\}\} \cup \{\emptyset, \{a\}, \{b\}, \{a, b\}\} \cup \{*\}$ of cardinality 6. The other similar sets are strictly smaller and the constructed term has width 6, which is not optimal. □

Hence, the construction of Algorithm 12 looks less interesting than that of Theorem 11, but it gives better bounds on clique-width in certain cases.

Proposition 13. *From a clean tree-decomposition of width k, Algorithm 12 produces a clique-width term of width at most $3 \cdot 2^{k-1}$ if the graph is undirected and at most $7 \cdot 2^{2(k-1)}$ if it is directed.*

Proof. We examine carefully the types of the subterms of t_u as in the proof of Theorem 11. The only case to consider is that of $u \in V_G$ with sons $u_1, \ldots, u_p$. Since the given tree-decomposition (T, f) is clean, $u \in f(u_i)$ for each i. Furthermore, if $|f(u)| = k+1$, then $f^*(u) := f(u) - \{u\}$ contains, for each i, at least one vertex not in $f(u_i)$ (because otherwise, $f(u_i) = f(u) \cup \{u_i\}$ and $|f(u_i)| = k + 2$). We denote by $\widehat{u_i}$ such a vertex.

Consider now: $\pi(H(u_i) \oplus L_{i-1}) = \Omega(T_{\leq}(u_i), f^*(u_i)) \cup B_{i-1} \cup \{*\}$. We first assume that $|f(u)| = k + 1$. We have:

$$B_{i-1} = \bigcup_{1 \leq j \leq i-1} \Omega(T_{\leq}(u_j), f^*(u_j) - \{u\}) \subseteq \Omega(T_{<}(u), f^*(u))$$

but $f^*(u) \notin \Omega(T_{<}(u), f^*(u))$ because of the vertices $\widehat{u_1}, \ldots, \widehat{u_p}$. Hence:

$$\pi(H(u_i) \oplus L_{i-1}) \subseteq \Omega(T_{\leq}(u_i), f^*(u_i)) \cup \{*\} \cup \mathcal{P}(f^*(u)) - \{f^*(u)\}.$$

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

The sets in $\Omega(T_{\leq}(u_i), f^*(u_i))$ cannot contain $\widehat{u_i}$, hence $f^*(u) \notin \Omega(T_{\leq}(u_i), f^*(u_i))$. The sets in $\Omega(T_{\leq}(u_i), f^*(u_i))$ that are not in $\mathcal{P}(f^*(u)) - \{f^*(u)\}$ must contain u, hence, are of the form $\{u\} \cup d$ for $d \in \mathcal{P}(f^*(u) - \{\widehat{u_i}\})$ and so, there are at most 2^{k-1} such sets. As $|\mathcal{P}(f^*(u)) - \{f^*(u)\}| = 2^k - 1$, we have:

$$|\pi(H(u_i) \oplus L_{i-1})| \leq 2^{k-1} + 2^k - 1 + 1.$$

If $|f(u)| \leq k$, then $B_{i-1} \subseteq \mathcal{P}(f^*(u))$ and so, $|B_{i-1}| \leq 2^{k-1}$. The sets in $\Omega(T_{\leq}(u_i), f^*(u_i))$ that are not in $\mathcal{P}(f^*(u))$ must contain u, hence, there are at most 2^{k-1} such sets. We obtain $|\pi(H(u_i) \oplus L_{i-1})| \leq 2^{k-1} + 2^{k-1} + 1 < 2^{k-1} + 2^k$. Hence, all terms t_u have width bounded by $2^{k-1} + 2^k = 3 \cdot 2^{k-1}$. (This bound is due to [9] where a similar construction is sketched.)

We now consider the case where G is directed. We have similar inclusions. If $|f(u)| = k + 1$, we have:

$$B_{i-1} \subseteq \Omega(T_{<}(u), f^*(u)) \subseteq P \text{ where } P := \mathcal{P}(f^*(u))^2 - \{(f^*(u), d), (d, f^*(u)) \mid d \subseteq f^*(u)\}.$$

Hence, $|B_{i-1}| \leq (2^k - 1)^2$. The pairs (d, d') in $\Omega(T_{\leq}(u_i), f^*(u_i)) - P$ must contain u in $d \cup d'$. The number of such pairs is bounded by $3 \cdot 2^{2(k-1)}$ (because there are 2^{k-1} sets included in $f^*(u) - \{u_i\}$). We obtain thus the bound

$$3 \cdot 2^{2(k-1)} + (2^{2k} - 2 \cdot 2^k + 1) + 1 = 7 \cdot 2^{2(k-1)} - 2^{k+1} + 2 < 7 \cdot 2^{2(k-1)}.$$

If $|f(u)| = k$, we have $|B_{i-1}| \leq (2^{k-1})^2$ and the number of pairs in $\Omega(T_{\leq}(u_i), f^*(u_i)) - (\mathcal{P}(f^*(u)) \times \mathcal{P}(f^*(u)))$ is bounded again by $3.2^{2(k-1)}$. We obtain thus the bound

$$3 \cdot 2^{2(k-1)} + 2^{2(k-1)} + 1 < 7 \cdot 2^{2(k-1)}$$

which completes the proof. □

For sake of completeness, we give a third construction from [10] that applies only to special tree-decompositions (cf. Definition 6). Its description is easier than that of that article.

Proposition 14. *If a graph G has special tree-width k, then $cwd(G) \leq k + 2$. A clique-width term witnessing this bound can be constructed in linear time from a special tree-decomposition of width k.*

Proof. Let (T, f) be a special tree-decomposition of a graph G. We can assume it is normal and $V_G = N_T$. The proof is the same for directed and undirected graphs. We will use the set of labels $C := \{\bot\} \uplus V_G$. For each $u \in N_T$, we define $K(u) := G[T_{\leq}(u) \cup f(u)]$ and we label its vertices as follows:

$\pi(w) :=$ `if` $N_G(w) \subseteq T_{\leq}(u)$ `then` $\bot$ `else` w.

If $\pi(w) \neq \bot$ then $w \in f(u)$. These graphs satisfy the following inductive characterization:

(1) If u is a leaf, then $K(u) = G[f(u)]$ with the labelling π. It can be defined by a term t_u of width $|f(u)| \leq k + 1$.

(2) Otherwise, u has sons $u_1, \ldots, u_p$, and the sets $T_{\leq}(u_i) \cup f(u_i)$ are pairwise disjoint because (T, f) is special. We let $\{v_1, \ldots, v_q\} := f(u) - (f(u_1) \cup \ldots \cup f(u_p))$, $q \geq 0$; these vertices are not in $K(u_1) \oplus \ldots \oplus K(u_p)$. We have:

$$K(u) = relab_h(A(K(u_1) \oplus \ldots \oplus K(u_p) \oplus \boldsymbol{v}_1(v_1) \oplus \ldots \oplus \boldsymbol{v}_q(v_q))),$$

where:

A is a composition of the edge additions that create the edges of $K(u)$ not in $K(u_1) \oplus \ldots \oplus K(u_p)$, and

h maps v to $\bot$ for all $v \neq \bot$ such that $\pi(v) = \bot$ (in $K(u)$).

We now bound the width of t_u in these last two cases.

$$\pi(K(u)) \subseteq f(u) \cup \{\bot\},$$

$$\pi(K(u_1) \oplus \ldots \oplus K(u_p) \oplus \boldsymbol{v}_1(v_1) \oplus \ldots \oplus \boldsymbol{v}_q(v_q)) \subseteq f(u) \cup \{\bot\}.$$

Hence, $cwd(G) \leq k + 1 + 1 = k + 2$. □

Remark 15. (1) As in Theorem 11 and Algorithm 12, we do not use the full power of the edge addition operations. The operations in A create edges with both ends in the set $f(u)$ that has cardinality at most $k + 1$.

(2) If the given decomposition (T, f) is not special, we can denote G by a term built with the operation of *parallel composition:* for edge disjoint graphs H and K (i.e., $E_H \cup E_K = \emptyset$), $H//K := (V_H \cup V_K, E_H \uplus E_K)$.

The graphs $G[f(u)]$ (without vertex labels) are defined inductively similarly as above. If u has sons $u_1, \ldots, u_p$, then:

$$G[f(u)] = G[f(u_1)]//\ldots//G[f(u_p)]//A_u$$

where A_u consists of the edges and vertices of $G[f(u)]$ not in $G[f(u_1)]//\ldots// \ G[f(u_p)]$. In a subsequent step, similar to the algorithm of Proposition 8, we can allocate "source" labels in $[k + 1]$ to convert the term that defines $G[f(u)]$ into a term of the "HR graph algebra" of [16], Chapter 2.

4. Sparse graphs

We apply Theorem 11 to several classes of sparse graphs. We recall from Section 1 that all graphs are simple.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

4.1. Planar graphs

Smoothing a vertex of degree 2 that has neighbours y and z in an undirected graph means replacing it and its two incident edges by a single edge between y and z, and then, fusing the parallel edges that may result. This transformation preserves planarity. We recall that a simple planar graph is uniformly 3-sparse.

Lemma 16. *Let $k \geq 3$. Let G be planar and $X, Y \subseteq V_G$ be such that $X \subseteq Y^c$ and $|Y| \leq k$.*
(1) *If G is undirected, then $|\Omega(X, Y)| \leq 6k - 9$.*
(2) *If G is directed, then $|\Omega(X, Y)| \leq 32k - 57$.*
If $k \leq 2$, the upper bounds are respectively 4 and 13.

Proof. (1) This assertion is Proposition 11 of [26]. We prove it for completeness and in order to prove the corresponding assertion about directed graphs. We consider disjoint sets X and Y, with Y of cardinality k, and we bound the number $|\Omega(X, Y)|$, *i.e.* the number of sets of the form $N_G(x) \cap Y$ for some $x \in X$.

We will bound $|\Omega(X, Y)|$ for graphs having edges between X and Y only. This suffices because removing the other edges and the vertices in $X^c - Y$ preserves planarity and does not modify $\Omega(X, Y)$.

We denote by X_1, X_2 and X_3 the sets of vertices of X having degree, respectively, at most 1, exactly 2 and at least 3. We have $|\Omega(X_1, Y)| \leq k + 1$.

Next we consider the vertices in X_2. We remove from G the vertices in $X - X_2$. We obtain a planar graph G'. By smoothing its vertices from X_2, we get a planar graph H with vertex set Y of cardinality k. Each edge of H corresponds to a set in $\Omega(X_2, Y)$. Hence, $|\Omega(X_2, Y)| = |E_H| \leq 3k - 6$.

We now consider the vertices in X_3. We remove from G the vertices in $X - X_3$. We get a bipartite planar graph K with edges between $V_K = X_3$ and Y. As each vertex in X_3 has degree at least 3 in K, we have $3\,|X_3| \leq |E_K|$. As K is planar and bipartite, $|E_K| \leq 2\,|V_K| - 4$. Hence, $3\,|X_3| \leq |E_K| \leq 2(|X_3| + k) - 4$ which gives $|X_3| \leq 2k - 4$, and so, $|\Omega(X_3, Y)| \leq |X_3| \leq 2k - 4$. Hence, $|\Omega(X, Y)| = |\Omega(X_1, Y)| + |\Omega(X_2, Y)| + |\Omega(X_3, Y)| \leq k + 1 + 3k - 6 + 2k - 4 = 6k - 9$.

(2) Assume now that G is directed. The undirected graph $Und(G)$ is obtained from G by forgetting edge directions and fusing any two parallel edges. We define X_1, X_2 and X_3 as above with degrees evaluated in $Und(G)$. Each edge between u and v in $Und(G)$ can come from three types of edges in G: $u \to v, v \to u$ and two opposite edges between u and v. Hence, $|\Omega(X_1, Y)| \leq 3k + 1$. By this observation, $|\Omega(X_2, Y)|$ is at most 9 times the corresponding value in $Und(G)$, hence $|\Omega(X_2, Y)| \leq 9(3k - 6)$. The above proof for an undirected graph shows that $2k - 4$ bounds $|X_3|$ hence $\Omega(X_3, Y)$. Hence, we get $|\Omega(X, Y)| \leq 3k + 1 + 9(3k - 6) + 2k - 4 = 32k - 57$.

The bounds $3k - 6$ (resp. $2k - 4$) on numbers of edges of simple planar graphs (resp. simple planar bipartite graphs) are valid if $k \geq 3$. Otherwise, inspecting the proofs yields the bounds $1 + 2 + 1 = 4$ for undirected graphs and $1 + 3 + 9 = 13$ for directed graphs. □

Theorem 17. *The clique-width of a simple planar graph of tree-width $k \geq 2$ is at most $32k - 24$ if it is directed, and at most $6k - 2$ if it is undirected.*

Proof. We apply Lemma 16 and Theorem 11, by noting that each set $f(u)$ has at most $k + 1$ elements. We get the bounds $32(k + 1) - 57 + 1 = 32k - 24$ on the clique-width of a directed graph and $6(k + 1) - 9 + 1 = 6k - 2$ for an undirected one. □

It follows from this result and Theorem 10(2) (a result from [28]) that clique-width and tree-width are linearly related.

Related work. By using the fact that the rank-width of an undirected graph is at most its tree-width plus 1 (proved in [30]), the article [26] establishes that the clique-width of a planar undirected graph is bounded by $12 \cdot twd(G) + 11$.

It proves also that, if G, undirected, is embeddable in a surface of Euler genus r (*i.e.*, a sphere with h handles and $r - 2h$ crosscaps) the bounds $3k-6$ and $2k-4$ in the proof of Lemma 16(1) are replaced by $3k-6+3r$ and $2k-4+2r$ respectively. The corresponding modifications of Lemma 16(2) and Theorem 17 give the bounds $cwd(G) \leq 32 \cdot twd(G) + O(r)$ for G directed and $cwd(G) \leq 6 \cdot twd(G) + O(r)$ for G undirected, where in both cases, G is embedded on some surface of genus r.

4.2. Uniformly q-sparse graphs

We recall from Section 1 that $\gamma(k, q)$ denotes the number of subsets of $[k]$ of cardinality at most q. It is $O(k^q)$ for fixed q and bounded by $k^q/(q - 1)!$ if $1 < q < k/2$. We will use $\gamma(k, q)$ for "small", fixed values of q.

Lemma 18. *Let $k \geq q > 1$ and G be uniformly q-sparse. Let $X, Y \subseteq V_G$ be such that $X \subseteq Y^c$ and $|Y| \leq k$.*
(1) *If G is undirected, then $|\Omega(X, Y)| \leq q \cdot k + \gamma(k, q)$.*
(2) *If G is directed, then $|\Omega(X, Y)| \leq q \cdot k + 3^q\gamma(k, q)$.*

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

Proof. Let $k \geq q > 1$ and G be uniformly q-sparse. We let X and Y be as in the proof of Lemma 16 and we bound $|\Omega(X, Y)|$.

(1) Let G be undirected and H be an orientation of G of indegree at most q (cf. Section 1). As in the proof of Lemma 16, we can assume that G and H are bipartite with edges between X and Y.

Let X_1 be the set of vertices $x \in X$ such that $N_H^+(x)$ is not empty. Since the orientation has indegree at most q and $N_H^+(x) \subseteq Y$, $|X_1| \leq q \cdot k$ and hence, $|\Omega(X_1, Y)| \leq |X_1| \leq q \cdot k$. ($\Omega$ is relative to G). For each vertex x of $X_2 := X - X_1$, we have $N_H^+(x) = \emptyset$ and $N_H^-(x)$ is a subset of Y of cardinality at most q. There are at most $\gamma(k, q)$ such sets, hence, $|\Omega(X_2, Y)| \leq \gamma(k, q)$. We get the claimed upper-bound since $|\Omega(X, Y)| \leq |\Omega(X_1, Y)| + |\Omega(X_2, Y)|$.

(2) We apply this argument to $Und(G)$ that is uniformly q-sparse. We still have $|\Omega(X_1, Y)| \leq q \cdot k$ but $|\Omega(X_2, Y)| \leq 3^q \gamma(k, q)$ as each edge of H can arise from three configurations of edges in G. □

Theorem 19. *For each $q \geq 1$, if G is uniformly q-sparse, then $cwd(G) = O(twd(G)^q)$.*

Proof. Immediate consequence of Lemma 18 and Theorem 11. □

We get $cwd(G) = O(twd(G)^{\lceil d/2 \rceil})$ for graphs of degree at most d (where the constant depends on d), but a proof similar to that of Theorem 19 gives $cwd(G) < d(twd(G) + 1) + 2$. Since planar graphs are uniformly 3-sparse, we get $cwd(G) = O(twd(G)^3)$ for them, but Theorem 17 also gives linear upper-bounds.

Related work. Theorem 21 of [26] proves that for every (fixed) r, we have $cwd(G) = O(\gamma(twd(G), r)) = O(twd(G)^r)$ for every undirected graph G in $\mathcal{S}_r$ (the class of graphs G having no subgraph isomorphic to $K_{r,r}$). As every uniformly q-sparse undirected graph G belongs to $\mathcal{S}_{2q+1}$, we deduce that, if G is uniformly q-sparse, then $cwd(G) = O(twd(G)^{2q+1})$ (for fixed q). However, the bound of Theorem 19 is better.

5. Bipartite graphs and hypergraphs

Bipartite graphs are interesting for many reasons. In particular, they can encode incidence graphs and hypergraphs as we will see, and also satisfiability problems for propositional formulae [25].

A bipartite graph G is *d-bounded* if all vertices of one of the two parts of V_G have degree at most d. For such a graph, $Und(G)$ has an orientation of indegree at most d, hence Theorem 19 gives $cwd(G) = O(twd(G)^d)$. We will improve this bound.

5.1. Hypergraphs as bipartite graphs

Definition 20. Hypergraphs and their tree-decompositions.

(a) A *hypergraph* is a triple $H = (V_H, E_H, inc_H)$ such that V_H and E_H are disjoint nonempty sets and $inc_H \subseteq V_H \times E_H$; V_H is the set of *vertices*, E_H is the set of *hyperedges* and $(v, e) \in inc_H$ means that v is *a vertex of* e (we also say that e is *incident to* v). In order to avoid uninteresting technical details, we assume that each hyperedge has at least one vertex and that each vertex belongs to some hyperedge. A hypergraph is a *q-hypergraph* if its hyperedges have at most q vertices. The directed bipartite graph *associated with* H is $Bip(H) := (V_H \cup E_H, inc_H)$ and H can be reconstructed from $Bip(H)$. If H is a q-hypergraph, then $Bip(H)$ is q-bounded. We also define the undirected graph $K(H)$ with set of vertices V_H and edges between any two vertices belonging to some hyperedge.

(b) A *tree-decomposition* of a hypergraph H is a pair (T, f) as for graphs with the condition that each hyperedge must have all its vertices in some set $f(u)$. Equivalently, (T, f) is a tree-decomposition of $K(H)$ because, for any tree-decomposition (T', g) of a graph, each clique of this graph is contained in some set $g(u)$. The *width* of (T, f) and the *tree-width* $twd(H)$ of H together with the notions of normal, clean and quasi-normal tree-decompositions are as for $K(H)$.

Fig. 3 shows the graph $G = Bip(H)$ associated with a 3-hypergraph H with hyperedges t, u, v, w, x, y, z and the tree T of a tree-decomposition (T, f) of H of width 2; the function f is defined in the following table ($s \in N_T$). In the figure, hyperedges are circled, and the edges of $Bip(H)$ are undirected.

s	$f(s)$	s	$f(s)$
a	a	g	g, e
b	b, a	h	h, e
c	c, b, a	i	i, h, e
d	d, c, a	j	j, e
e	e, c, a		

Lemma 21. (1) *For every hypergraph H, $twd(Bip(H)) \leq twd(H) + 1$ but $twd(H)$ is not bounded in $twd(Bip(H))$.*

(2) *If H is a q-hypergraph, then $twd(H) \leq q(twd(Bip(H)) + 1) - 1$.*

Proof sketch. (1) Let (T, f) be a tree-decomposition of H. For each hyperedge e, there is a node u of T such that all vertices of e are in $f(u)$ and we add to T a new son u' of u with associated set $\{e\} \cup f(u)$. We get a tree-decomposition of $Bip(H)$ of

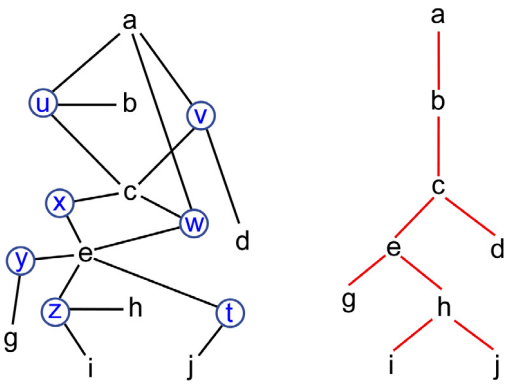

Fig. 3. *Bip*(H) and the tree T of a tree-decomposition of H.

width $twd(Bip(H))+1$. One cannot bound $twd(H)$ in terms of $twd(Bip(H))$ alone: if H has one hyperedge with $n+1$ vertices, we have $twd(Bip(H))=1$ and $twd(H)=n$.

(2) Let (T,f) be a tree-decomposition of $Bip(H)$ of width k such that H is a q-hypergraph. We define

$f'(u) := f(u) \cup \{x \in V_H \mid inc_H(x,e) \text{ for some } e \in f(u) \cap E_H\} - (f(u) \cap E_H)$.

Then (T,f') is a tree-decomposition of H and $|f'(u)| \leq q\,|f(u)| \leq q(k+1)$ which yields the result. (This result is a remark after Theorem 5.4 in [29].) □

We do not have $twd(Bip(H)) \leq twd(H)$ in general: let H be the hypergraph with 3 vertices and 3 hyperedges containing all these vertices. Then, $twd(H)=2$ but $twd(Bip(H))=3$ (because $Bip(H)$ contains K_4 as a minor).

This lemma shows that for fixed q, the tree-width of a q-hypergraph and that of its associated bipartite graph are linearly related. Theorem 19 shows that $cwd(Bip(H)) = O(twd(Bip(H))^q)$ for a q-hypergraph. We have $cwd(Bip(H)) = O(twd(H)^q)$ by this fact and Lemma 21, but we can do better.

Theorem 22. *Let $q \geq 2$. For every q-hypergraph H, we have:*

$$cwd(Bip(H)) \leq \gamma(twd(H)+1, q-1)+1 = O(twd(H)^{q-1}).$$

Proof. Let (T,f_T) be a normal tree-decomposition[15] of H, *i.e.* of $K(H)$, of width $k = twd(H)$. The vertices of a hyperedge e are in $f_T(u)$ for some node u of T, hence are linearly ordered by $\leq_T$ because (T,f_T) is normal; we let $\widehat{e}$ be the smallest one.

We extend T into a tree U with set of nodes $N_T \cup E_H$ as follows. For each $u \in V_H$, we let $e_1,\ldots,e_m$ be the hyperedges e such that $\widehat{e}=u$; we replace the edge $u - p_T(u)$ of T by the path $u - e_1 - \cdots - e_m - p_T(u)$; if $m=0$ we do nothing; if u is the root, we put the path $e_1 - \cdots - e_m$ above u with e_m as new root (these hyperedges have u as unique vertex). The vertices of a hyperedge e are $\widehat{e}$ that is below it (in U), and, at most $q-1$ vertices that are above.

Fig. 4 shows $Bip(H)$ and the tree U for H and T of Fig. 3. The edges of U not in $Bip(H)$ are shown with dotted lines. The nodes for hyperedges w and x are inserted between e and c. We could have inserted x below w.

It is clear that U is a normal tree for $Bip(H)$. We obtain a normal tree-decomposition (U,f_U) of $Bip(H)$. (It is not the tree-decomposition constructed in the proof of Lemma 21(1); its width is not bounded in terms of k: just consider several parallel edges between two vertices.) In order to use Theorem 11, we bound the cardinalities of the sets $\Omega(U_{<}(w), f_U(w))$.

If $w \in V_H$, then $\Omega(U_{<}(w), f_U(w))$ consists of the following sets:

first, the sets $N_{Bip(H)}(u) \cap f_U(w)$ for $u \in V_H$, $u <_U w$; these sets are empty, because the neighbours in $Bip(H)$ of such u are hyperedges e such that $e <_T u$ or $\widehat{e}=u$ but, in both cases, $e <_T w$, hence $e \notin f_U(w)$;

second, the sets $N_{Bip(H)}(e) \cap f_U(w)$ for $e \in E_H$, $e <_U w$; these sets are subsets of $f_T(w)$ of cardinality at most $q-1$, because they are the sets of ends $v \geq_T w$ of the edges of $K(H)$ whose other end is $\widehat{e} <_U e$; (in the example of Fig. 4, for $w := c$, the sets in $\Omega(U_{<}(w), f_U(w))$ are $\emptyset$, $\{c\}$ and $\{a,c\}$).

Hence, $|\Omega(U_{<}(w), f_U(w))| \leq \gamma(k+1, q-1)$.

If $w = e \in E_H$, then $\Omega(U_{<}(w), f_U(w))$ consists of the following sets:

first, the sets $N_{Bip(H)}(u) \cap f_U(e)$ for $u \in V_H$, $u <_U e$: if $u = \widehat{e}$, then $N_{Bip(H)}(u) \cap f_U(e) = N(e) := \{e_1 \in E_H \mid e \leq_U e_1, \widehat{e_1} = \widehat{e}\}$, otherwise, $u <_T \widehat{e}$, and the neighbours of u are hyperedges $e_1 <_U \widehat{e} <_U e$, hence, $N_{Bip(H)}(u) \cap f_U(e) = \emptyset$; (in the example of Fig. 4, we have $N(w^O) = \{w^O, x^O\}$ where w^O denotes the "circled w" and similarly for x^O; we have $N(x^O) = \{x^O\}$);

second, the sets $N_{Bip(H)}(e_1) \cap f_U(e)$ for $e_1 \in E_H$, $e_1 <_U e$: these sets are subsets of $f_T^*(\widehat{e})$ of cardinality at most $q-1$, because they are the sets of ends $v \geq e > \widehat{e}$ of the edges of $K(H)$ whose other end is below e, hence below $\widehat{e}$ or equal to it; (in the example of Fig. 4, we have $N_{Bip(H)}(w^O) \cap f_U(x^O) = \{a,c\}$).

[15] The mapping f_T is "minimal", cf. Definitions 1(c).

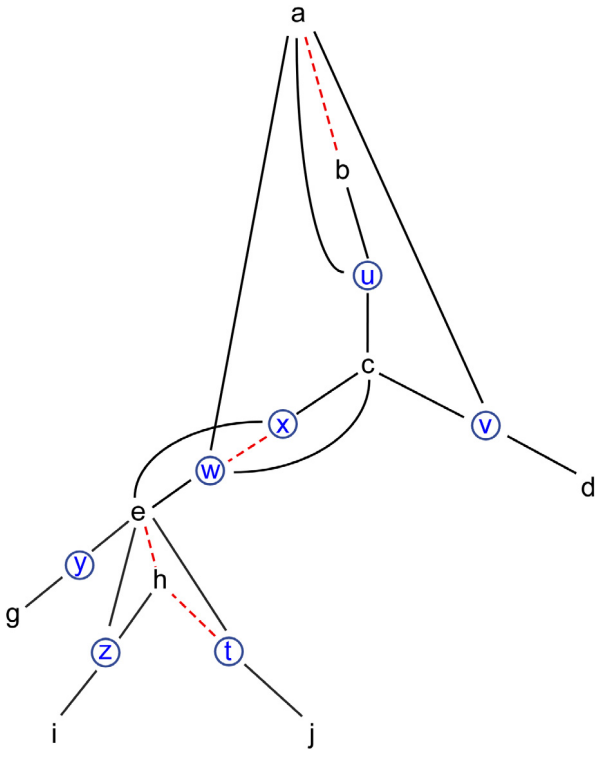

Fig. 4. Tree U for $Bip(H)$ of Fig. 3.

Hence, $|\Omega(U_{<}(w), f_U(w))| \leq 1 + \gamma(k, q-1) \leq \gamma(k+1, q-1)$. The claimed result[16] follows then from Theorem 11. □

5.2. Incidence graphs

Incidence graphs are useful for the logical expression of graph properties in MSO logic, as we will recall from [10–12,16] in the next section.

Let G be undirected, possibly with loops and parallel edges. Its edge set is E_G. Its *incidence graph* $Inc(G)$ is the directed bipartite graph $(V_G \cup E_G, inc_G)$ such that $inc_G := \{(v, e) \in V_G \times E_G \mid v \text{ is a vertex of } e\}$. A loop has degree one in $Inc(G)$. If G has no loop, then $Und(Inc(G))$, the undirected graph underlying $Inc(G)$ is obtained from G by adding a new (degree 2) vertex on each edge. If G is considered as a 2-hypergraph, then $Inc(G) = Bip(G)$.

If G is directed, then $Inc(G)$ is defined as $(V_G \cup E_G, inc_G)$ with $inc_G := \{(v, e) \in V_G \times E_G \mid e : v \rightarrow_G w \text{ for some vertex } w\} \cup \{(e, v) \in E_G \times V_G \mid e : w \rightarrow_G v \text{ for some vertex } w\}$.

Tree-width and clique-width for G and $Inc(G)$ are related as follows. We have $twd(Inc(G)) = twd(G)$ except if G is a forest where at least one edge is replaced by several parallel edges: in that case, $twd(G) = 1$, $twd(Inc(G)) = 2$, $cwd(Inc(G)) \leq 3$ if G is undirected and $cwd(Inc(G)) \leq 4$ if G is directed.

By Theorem 10(2), we have $twd(Inc(G)) \leq 6 \cdot cwd(Inc(G)) - 1$. The following corollary of Theorem 22 is proved in [7] in a different way.

Corollary 23. *We have $cwd(Inc(G)) \leq twd(G) + 3$ if G is undirected and $cwd(Inc(G)) \leq 2 \cdot twd(G) + 4$ if it is directed.*

Proof. If G is undirected, Theorem 22 yields that the clique-width of $Inc(G) = Bip(G)$ is bounded by $\gamma(twd(G)+1, 1)+1 = twd(G)+3$.

If G is directed, we construct U as in the proof of Theorem 22. In that case, every edge e of G that is not a loop links a vertex $\widehat{e}$ below it in U and one above it. If it is a loop, then $\widehat{e} \rightarrow e$ and $e \rightarrow \widehat{e}$. We use the notation of the proof of Theorem 11 for directed graphs. The sets $\Omega(U_{<}(w), f_U(w))$ consist of the pairs $(N^{+}_{Inc(G)}(u) \cap f_U(w), N^{-}_{Inc(G)}(u) \cap f_U(w))$ for $u <_U w$.

16 A similar result in [7] states that $cwd(S(H)) = O(twd(H)^{q-1})$ if H is a q-hypergraph and $S(H)$ is $Bip(H)$ augmented with undirected edges between any two vertices of H.

If $w \in V_G$, $\Omega(U_<(w), f_U(w))$ consists of $(\emptyset, \emptyset)$ and pairs $(v, \emptyset)$ or $(\emptyset, v)$ for some $v \in f_U(w)$. Hence, $|\Omega(U_<(w), f_U(w))| \leq 1 + 2(k+1)$.

If $w = e \in E_G$, then $\Omega(U_<(w), f_U(w))$ consists of $(\emptyset, \emptyset)$, the pair $(N^+_{Inc(G)}(\widehat{e}) \cap f_U(e), N^-_{Inc(G)}(\widehat{e}) \cap f_U(e))$ and pairs $(v, \emptyset)$ or $(\emptyset, v)$ for some $v \in f_T(\widehat{e}) - \{\widehat{e}\}$. Hence, $|\Omega(U_<(w), f_U(w))| \leq 1 + 1 + 2k < 2k + 3$. We obtain the bound $2k + 4$ by Theorem 11. □

Hence, tree-width and $cwd(Inc(\cdot))$ are linearly equivalent measures.

Remark 24. The empty set (or the pair $(\emptyset, \emptyset)$) is used in the construction of a term that denotes $Inc(G)$ as a label for its vertices in V_G as well as in E_G. In view of application to the model-checking of MSO_2 properties (see Section 6 and [11,12]), it is useful to distinguish, in the term that defines $Inc(G)$, the labels for the vertices of G from those that define its edges (that are also vertices of $Inc(G)$). For this purpose, we duplicate these "empty" labels (and no others). So, we can construct $Inc(G)$ with two labels for the vertices of G and $twd(G) + 2$ labels for its edges, *i.e.*, the vertices in $E_G \subseteq V_{Inc(G)}$. If G is directed these figures are respectively 2 and $2 \cdot twd(G) + 3$.

6. Algorithmic applications

We discuss some applications of our constructions to the verification of *monadic second-order expressible* graph properties (*MSO properties* in short) by means of automata that process clique-width terms denoting the input graphs. This method is implemented in the running system AUTOGRAPH.[17] This section being informal, we will use examples and we refer the reader to [13,16] for definitions.

6.1. Model-checking with fly-automata

We give the example of a monadic second-order (MSO) sentence[18] expressing that a graph G, defined as the relational structure $\langle V_G, edg_G \rangle$, is 3-colourable. This sentence is $\exists X, Y.Col(X, Y)$ where $Col(X, Y)$ is the formula

$$X \cap Y = \emptyset \wedge \forall u, v.\{edg(u, v) \Longrightarrow [\neg(u \in X \wedge v \in X) \wedge \neg(u \in Y \wedge v \in Y) \wedge \neg(u \notin X \cup Y \wedge v \notin X \cup Y)]\},$$

expressing that X, Y and $V_G - (X \cup Y)$ are the three colour classes of a proper 3-colouring of the considered graph G.

An MSO sentence intended to express a graph property can only use quantifications over vertices and sets of vertices. More powerful are the *MSO_2 sentences*, that can also use quantifications over edges and sets of edges. We recall the following "algorithmic meta-theorem" [16,18,20,21].

Theorem 25. (a) *For every integer k and every MSO sentence φ, there exists a linear-time algorithm that checks the validity of φ in any graph given by a term in $T(F_{[k]})$, whence of clique-width at most k. The computation time is linear in the size of the term.*

(b) *For every integer k and every MSO_2 sentence φ, there exists a linear-time algorithm that checks the validity of φ in any graph given by a tree-decomposition of width at most k. The computation time is linear in the size of the tree-decomposition.*

Assertion (b)[19] is actually a consequence of (a) because:

(1) an MSO_2 property of a graph G is nothing but an MSO property of its incidence graph $Inc(G)$,

(2) if G has tree-width k, then $Inc(G)$ has clique-width at most $f(k)$ for some fixed linear function f (cf. Corollary 23), and

(3) a tree-decomposition of G of width k can be converted in linear time (for fixed k) into a clique-width term of width at most $f(k)$ that defines $Inc(G)$.

Point (1) is just a matter of definitions. Point (2) and the linear-time transformation of (3) make practically useable this reduction of (b) to (a). This observation is developed in [11,12]. MSO_2 sentences are more expressive than MSO ones, but bounded tree-width is necessary for having FPT algorithms to check the corresponding properties in this way, via incidence graphs.

Some linear-time algorithms intending to implement (a) use finite automata that take as input terms t in $T(F_{[k]})$ and answer whether the graph $G(t)$ satisfies the considered property. However, these automata are much too large to implementable in the classical way by means of transition tables. To the opposite, *fly-automata* (*FA* in short) *compute* the transitions that are needed during the run on a given term and thus overcome the size obstacle.

We review FA informally. Let C be a countably infinite set of labels. A *deterministic fly-automaton* $\mathcal{A}$ over F_C has a possibly infinite set of states $Q_{\mathcal{A}} \subseteq (C \uplus B)^*$ where B is a finite set of auxiliary symbols, typically *True*, 0, 1, (,), {, } ,",", etc. Its transitions are of the forms $\mathbf{a} \rightarrow_{\mathcal{A}} p$, $f[q] \rightarrow_{\mathcal{A}} p$ and $\oplus[q, q'] \rightarrow_{\mathcal{A}} p$, where $a \in C, f \in F_C$ is unary, $q, q', p \in Q_{\mathcal{A}}$ and p is defined

[17] AUTOGRAPH can even compute values associated with graphs [15], for an example, the number of 3-colourings. It is written in Common Lisp by I. Durand. See http://dept-info.labri.u-bordeaux.fr/~idurand/autograph.

[18] A *sentence* is a logical formula without free variables.

[19] By a result of Bodlaender (see [5,20,21]), a tree-decomposition of G of width k can be computed in linear time if there exists one. Hence the variant of (b) where a tree-decomposition is not given but must be computed also holds, but this variant is not a consequence of (a). Furthermore, the linear time decomposition algorithm is not practically implementable. See [4] for useable algorithms.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

in a unique way by an algorithm (that is part of the definition of $\mathcal{A}$) from a, or from f and q, or from q and q'. The (possibly infinite) set $Acc_{\mathcal{A}} \subseteq Q_{\mathcal{A}}$ of accepting states is decided by an algorithm. It follows that, on each term t, the automaton $\mathcal{A}$ has a unique (bottom-up) run. This run is computable and so is $q_{\mathcal{A}}(t)$, the (unique) state reached at position ε (the root of the syntactic tree of t). Hence, the membership of t in $L(\mathcal{A})$, the language accepted by $\mathcal{A}$, is decidable.

The time computation of a deterministic FA $\mathcal{A}$ on a term t is $\Sigma\{\tau_{\mathcal{A}}(u) \mid u \in Pos(t)\}$ where $\tau_{\mathcal{A}}(u)$ is the time taken to compute the state p reached at position u by the run of $\mathcal{A}$, plus the time taken to check whether $q_{\mathcal{A}}(t) \in Acc_{\mathcal{A}}$.

For model-checking, we are interested in cases where $t \in L(\mathcal{A})$ if and only if $G(t)$ satisfies the property to check. Note that the same automaton, hence, the same algorithm, works for graphs of any clique-width as C is infinite.

Example 26. A deterministic fly-automaton $\mathcal{A}$ that checks 3-colourability.

Let $\Gamma := \{1, 2, 3\}$ be the set of colours. Let G be a C-labelled graph. For each mapping $\gamma : V_G \to \Gamma$, we define $\widetilde{\gamma} := \{(a, i) \in C \times \Gamma \mid \gamma(x) = i$ for some a-labelled vertex $x\}$.

We define $\xi(G)$ as the finite set of finite sets $\widetilde{\gamma}$ such that γ is a proper 3-colouring of G (no two adjacent vertices have the same colour). For $t \in T(F_C)$, we define $\xi(t) := \xi(G(t))$. Clearly, $\xi(t)$ can be written as a word over $C \uplus \Gamma \uplus A$ where A is the alphabet consisting of (,),{,} and ",". The function ξ satisfies the following inductive property:

$$\xi(\mathbf{a}) = \{\{(a, 1)\}, \{(a, 2)\}, \{(a, 3)\}\} \text{ for } a \in C,$$
$$\xi(add_{a,b}(t)) = \{\alpha \in \xi(t) \mid \text{ there is no } i = 1, 2, 3 \text{ such that } (a, i) \text{ and } (b, i) \text{ belong to } \alpha\},$$
$$\xi(relab_h(t)) = h(\xi(t)) \text{ where } h \text{ replaces in the word } \xi(t) \text{ each label } a \in C \text{ by } h(a),$$
$$\xi(t_1 \oplus t_2) = \{\alpha \cup \beta \mid \alpha \in \xi(t_1), \beta \in \xi(t_2)\}.$$

These properties give the transitions of the desired FA $\mathcal{A}$ whose set of states is $\mathcal{P}(\mathcal{P}(C \times \Gamma))$, identified to a language over $C \uplus \Gamma \uplus A$, and such that $q_{\mathcal{A}} = \xi$. The transitions are:

$$\mathbf{a} \to_{\mathcal{A}} \{\{(a, 1)\}, \{(a, 2)\}, \{(a, 3)\}\},$$
$$add_{a,b}[q] \to_{\mathcal{A}} \{\alpha \in q \mid \text{ there is no } i = 1, 2, 3 \text{ such that } (a, i) \text{ and } (b, i) \text{ belong to } \alpha\},$$
$$relab_h[q] \to_{\mathcal{A}} h(q),$$
$$\oplus[q, q'] \to_{\mathcal{A}} \{\alpha \cup \beta \mid \alpha \in q, \beta \in q'\}.$$

All states are accepting except the empty set. The set of all states that can occur in a run over a term in $T(F_{[k]})$ (assuming $[k] \subseteq C$) is finite but of cardinality $2^{2^{3k}}$. Hence, it cannot be listed in a table for useful values of k. □

We go back to the general case. We fix C. If φ is an MSO sentence, we denote by $L(\varphi)$ the set of terms $t \in T(F_C)$ such that $G(t) \models \varphi$. The proof of Theorem 25 (a) is based on an algorithm $\mathcal{MC}$ that constructs, from any φ, a deterministic FA $\mathcal{A}(\varphi)$ over F_C that recognizes the language $L(\varphi)$. However, in Example 26, we have constructed an FA "directly" from our understanding of 3-colourability, without using its expression by an MSO sentence.

Let $h : C \to C'$ be a bijection. It extends into a bijection $F_C \to F_{C'}$ (each label $a \in C$ occurring in a symbol of F_C is replaced by $h(a)$) and into a bijection $T(F_C) \to T(F_{C'})$; both are denoted by h. The deterministic FA $h(\mathcal{A}(\varphi))$ over $F_{C'}$, obtained from $\mathcal{A}(\varphi)$ by replacing f by $h(f)$ and each state q by $h(q)$ in each transition, is the one constructed by $\mathcal{MC}$ where we replace C by C'. We have $L(h(\mathcal{A}(\varphi))) = h(L(\mathcal{A}(\varphi)))$.

Theorem 27. *Let C be a countable set of vertex labels. There is an algorithm that constructs, for each MSO sentence φ, a deterministic FA $\mathcal{A}(\varphi)$ over F_C that recognizes the language $L(\varphi) \subseteq T(F_C)$ and satisfies the following properties, for all $t, t' \in T(F_C)$:*
(i) *$q_{\mathcal{A}(\varphi)}(t) \in (B \uplus \pi(t))^*$ where B is a finite set disjoint from C,*
(ii) *if $G(t)$ is isomorphic to $G(t')$, then $q_{\mathcal{A}(\varphi)}(t) = q_{\mathcal{A}(\varphi)}(t')$,*
(iii) *if $h : C \to C'$ is a bijection and $B \cap C' = \emptyset$, then $q_{h(\mathcal{A}(\varphi))}(h(t)) = h(q_{\mathcal{A}(\varphi)}(t))$.*

The proof is by induction on the structure of φ (an adequate inductive assertion is used for formulae with free variables; see [13,14]). It follows from (iii) that a set of labels C can be replaced by C' in bijection with C by h: the computation of $h(\mathcal{A}(\varphi))$ over $h(t)$ is the same as that of $\mathcal{A}(\varphi)$ over t, up to the replacement in the run of each label a by $h(a)$. However, a difference in the computation times of $h(\mathcal{A}(\varphi))$ and $\mathcal{A}(\varphi)$ may arise from the codings of labels. The computation time $\tau_{\mathcal{A}(\varphi)}(u)$ of a transition is bounded by $a_\varphi \cdot \theta \cdot \tau'_{\mathcal{A}(\varphi)}(u)$ where $\tau'_{\mathcal{A}(\varphi)}(u)$ is the number of comparisons of two labels during the computation of the state at a position u in terms of the states at its sons θ bounds the time taken for one comparison and a_φ depends only on φ. If $t \in T(F_{[k]})$ where k is "small", then, one can take $\theta = 1$. This may not be the same if C is "large", as in Theorem 11 and Algorithm 12. Hence, although FA can take as inputs terms in $T(F_C)$ for large sets C, this observation motivates the use of Proposition 8.

However, the algorithm of Proposition 8 does not build a bijection h from C to $\mathbb{N}_+$ making $t \in T(F_C)$ into an equivalent term $h(t)$ in $T(F_{[cwd(t)]})$. For each $t \in T(F_C)$ and each position u of t, it defines a term t_u, and a bijection $h_u : \pi(t/u) \to \pi(t_u) \subseteq \mathbb{N}_+$ such that $t_u \equiv relab_{h_u}(t/u)$. It follows from Assertions (ii) and (iii) of Theorem 27 that $q_{\mathcal{A}'(\varphi)}(t_u) = h_u(q_{\mathcal{A}(\varphi)}(t/u))$ for each $u \in Pos(t)$, where $\mathcal{A}'(\varphi)$ is the FA over $T(F_{\mathbb{N}_+})$ constructed from φ by algorithm $\mathcal{MC}$. The term t_u is, in most cases, larger than t/u because it is built from it by insertions of relabellings. However, these relabellings are bijective (see the next section) and the corresponding transitions are nothing but substitutions of symbols in the words that represent the states.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

Hence, to conclude, a sentence of φ can be checked by running $\mathcal{A}(\varphi)$ either on a term t over F_C where C may be much larger than the width of t, or on an equivalent term over $F_{[wd(t)]}$. Our first experiments with AUTOGRAPH seem to favour the second method.

6.2. Constructions of clique-width terms

The definition of clique-width terms given in Section 2.2 is appropriate for bounding clique-width. However, in concrete applications, some constraints on these terms are necessary or useful for limiting computation times of automata. Here below, we list them, and we explain how arbitrary terms can be transformed into equivalent ones that satisfy these constraints.

Constraints on clique-width terms as inputs of FA.

(1) The disjoint union operation $\oplus$ must take exactly two arguments. To ensure this, we replace any term of the form $t_1 \oplus \ldots \oplus t_n$ by $t_1 \oplus (t_2 \oplus (\ldots \oplus t_n)..)$ (cf. the proof of Proposition 8).

(2) All edge addition operations should be useful. There are two cases where an occurrence of such an operation can be removed. First, if in a term $\overrightarrow{add}_{a,b}(t)$, a or b is not in $\pi(t)$: the topmost occurrence of $\overrightarrow{add}_{a,b}$ has no effect and can be removed (the proof of Proposition 8 does that actually). The second case is when $G(t)$ contains an edge from an a-labelled vertex to a b-labelled one; then, we say that the term $\overrightarrow{add}_{a,b}(t)$ has a *redundancy*, and at least one edge addition operation can be removed from t. See [13] for details. These simplifications of terms apply to $add_{a,b}$ in similar ways.

(3) The transitions of an FA relative to a relabelling $relab_h$ may be uneasy to program if h is neither bijective nor elementary. We recall that a relabelling $relab_h$ is *bijective on a term* t, if h is injective on $\pi(t)$, hence is a bijection: $\pi(t) \to h(\pi(t))$. In this case, we have $q_{\mathcal{A}(\varphi)}(relab_h(t)) = h(q_{\mathcal{A}(\varphi)}(t))$. Hence, $relab_h[q] \to_{\mathcal{A}(\varphi)} h(q)$ if h is injective on the set of labels from C that occur in q and $h(q)$ is obtained by substituting everywhere in q each label a by $h(a)$, hence in a straightforward manner.

In the currently running version of AUTOGRAPH, transitions are defined for elementary relabellings, i.e., for those of the form $relab_{a\to b}$. Every term $relab_h(t)$ can be replaced by $relab_{h'}(R(t))$ where R is the composition of r elementary relabellings, h' is injective on $\pi(R(t))$ and r is the number of labels $a \in \pi(t)$ such that $h(a) = h(b)$ for some $b \in \pi(t)$ where b is before a in some fixed enumeration of C. Every relabelling can also be expressed as a composition of elementary ones at the cost of using at most one extra label.

6.3. Experiments

The constructions of Proposition 8, Theorems 11 and 25(a) have been implemented in AUTOGRAPH and give satisfactory results. The next step is the implementation of Theorem 25(b) via incidence graphs.

7. Conclusion

For uniformly q-sparse graphs, clique-width is polynomially bounded in terms of tree-width and we have algorithmically efficient transformations of quasi-normal tree-decompositions into clique-width terms witnessing the claimed upper-bounds. We also have linear bounds for planar graphs and incidence graphs. Applications to FPT graph algorithms for checking monadic second-order properties expressed with edge set quantifications are developed in [11,12].

In all our proofs that yield bounds on clique-width in terms of tree-width, the tree T of a given tree-decomposition (see Theorem 11, Algorithm 12 and Proposition 14) is also, up to a minor transformation in Algorithm 12, the syntactic tree of the constructed clique-width term. However, this is not the case for the bound of Assertion (1.2) in Theorem 10, because its proof is based on a difficult result of [32] that restructures the tree T in a complicated way.

We propose three open questions.

1. Let G be a graph given with an optimal tree-decomposition of width k and t be a clique-width term produced by one of the first two algorithms of Section 3. How large is $wd(t) - cwd(G)$? Is it polynomial in k? In words, how far from being optimal is the term t. We recall that our algorithms do not use the full power of the edge addition operations. In Example (3) after Theorem 11, we have $wd(t) - cwd(G) = twd(G) - 1$ for the tree-decomposition $(T'', f_{T''})$.

2. For fixed q, does there exist $p < q$ such that $cwd(G) = O(twd(G)^p)$ for every uniformly q-sparse graph G?

3. For which classes of graphs other than those of Theorems 10(1), 17 and 22, is clique-width linearly bounded in terms of tree-width?

Acknowledgements

I thank I. Durand, S. Oum, M. Kanté and the referees for their useful comments.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

Appendix

The following observations are intended to clarify some aspects of quasi-normal tree-decompositions.

Quasi-normal tree-decompositions and minors

We examine how quasi-normal tree-decompositions behave through the graph reductions that yield the minor quasi-order.

Let (T, f) be a quasi-normal tree-decomposition of a graph G. If H is a subgraph of G, then (T, f') where $f'(u) := f(u) \cap V_H$ is a quasi-normal tree-decomposition of H of no larger width than (T, f).

We now consider H obtained from G by the contraction of an edge between u and v. We assume that $v <_T u$ and we build concretely H as follows: we delete v and the edge (or edges) between u and v, and for every $w \neq u$, we make each edge between w and v into an edge between w and u (we preserve its direction if G is directed).

We first replace (T, f) by (T, f_T) that is quasi-normal and of no larger width. The tree T is quasi-normal for H. Let f'_T be the corresponding "minimal" mapping (cf. Definitions 1(c)), so that (T, f'_T) is a quasi-normal tree-decomposition of H. We examine its width.

The only vertices w for which the set $f'_T(w)$ might differ from $f_T(w)$ are those that are comparable with v with respect to $\leq_T$. We consider the different cases:

Case 1: $w <_T v$. If $v \in f_T(w)$, then $f'_T(w) = (f_T(w) - \{v\}) \cup \{u\}$, otherwise $f'_T(w) = f_T(w)$.

Case 2: $w = v$. Then $f'_T(w) = f'_T(v) \subseteq f^*_T(v)$.

Case 3: $v <_T w <_T u$. We have $u \in f_T(w)$ (because of the contracted edge); it follows that $f'_T(w) = f_T(w)$ because the only redirected edges that now "jump" over w reach u.

Case 4: $u \leq_T w$. The redirected edges that "jump" over w reaching $s >_T w$ arise from edges between v and s. It follows that $s \in f_T(w)$. Hence $f'_T(w) = f_T(w)$.

Only Case 1 yields a modification of f_T (that replaces v by u in each $f_T(w)$ for $w <_T v$) so that $|f'_T(w)| \leq |f_T(w)|$; we have $|f'_T(w)| = |f_T(w)| - 1$ if and only if $u \in f_T(w)$. The width of (T, f'_T) is no larger than that of (T, f_T) and differs by at most one.

Boolean-width.

We review boolean-width [8] because it is based on a function similar to our Ω. Let G be an undirected graph and X a set of vertices. As in this article, we define $UN(X) := \{N_G(Z) - X \mid Z \subseteq X\}$. Clearly, $\Omega(X, V_G - X) \subseteq UN(X)$. We also have $|UN(X)| = |UN(V_G - X)|$ and we define $bdim(X)$ as $\log_2(|UN(X)|)$.

Let T be a binary (rooted) tree whose leaves are the vertices of G. We define $bwd(T)$ as the maximum value of $bdim(T_{\leq}(u) \cap V_G)$ for a node u of T, and the *boolean-width* of G, denoted by $bwd(G)$, as the minimum of $bwd(T)$ over all such trees. The purpose of taking a logarithm is to have $0 \leq bwd(G) \leq |V_G|$.

The following facts are proved in [8] (Theorem 3):

(1) $\log_2(cwd(G)) - 1 \leq bwd(G) \leq cwd(G)$,

(2) there exist an infinite family of graphs G for which $bwd(G) = O(\log_2(cwd(G)))$,

(3) there exist an infinite family of graphs G for which $cwd(G) = O(bwd(G))$.

In particular, the same families of undirected graphs have bounded clique-width and bounded boolean-width, and the comparisons of Fact 1 are essentially optimal.

From Fact 1 and Theorem 9(2), we get $bwd(G) = O(2^{twd(G)})$. However, we have better:

(4) $bwd(G) \leq twd(G)^2/4 + O(twd(G))$

by Corollary 1 of [8] and the bound $twd(G) + 1$ of the rank-width of G established in [30]. We leave open the problem of designing a construction that witnesses Fact 4.

References

[1] S. Arnborg, D. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebr. Discrete Methods 8 (1987) 277–284.
[2] H. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, Theoret. Comput. Sci. 209 (1998) 1–45.
[3] H. Bodlaender, P. Drange, M. Dregi, F. Fomin, D. Lokshtanov, M. Pilipczuk, A $c^k n$ 5-approximation algorithm for treewidth, SIAM J. Comput. 45 (2016) 317–378.
[4] H. Bodlaender, A. Koster, Treewidth computations I. Upper bounds, Inform. and Comput. 208 (2010) 259–275.
[5] H. Bodlaender, S. Kratsch, V. Kreuzen, Fixed-parameter tractability and characterizations of small special treewidth, Proc. WG Lect. Notes Comput. Sci. 8165 (2013) 88–99.
[6] H. Bodlaender, S. Kratsch, V. Kreuzen, O. Kwon, S. Ok, Characterizing width two for variants of treewidth, Discrete Appl. Math. 216 (2017) 29–46.
[7] T. Bouvier, Graphes et décompositions (Doctoral dissertation), Bordeaux University, 2014.
[8] B. Bui-Xuan, J. Telle, M. Vatshelle, Boolean-width of graphs, Theoret. Comput. Sci. 412 (2011) 5187–5204.
[9] D. Corneil, U. Rotics, On the relationship between clique-width and tree-width, SIAM J. Comput. 34 (2005) 825–847.
[10] B. Courcelle, On the model-checking of monadic second-order formulas with edge set quantifications, Discrete Appl. Math. 160 (2012) 866–887.
[11] B. Courcelle, Fly-automata for checking monadic second-order properties of graphs of bounded tree-width, Electron. Notes Discrete Math. 50 (2015) 3–8. Proceedings of LAGOS 2015, Beberibe, Brazil; a long version is in http://www.labri.fr/perso/courcell/Conferences/BC-Lagos2015.pdf.
[12] B. Courcelle, Fly-automata for checking MSO_2 graph properties, Discrete Appl. Math. (2016). http://dx.doi.org/10.1016/j.dam.2016.10.018. https://hal.archives-ouvertes.fr/hal-01234622v2, in press.
[13] B. Courcelle, I. Durand, Automata for the verification of monadic second-order graph properties, J. Appl. Log. 10 (2012) 368–409.
[14] B. Courcelle, I. Durand, Fly-automata, model-checking and recognizability, *Proceedings of the workshop Frontiers of Recognizability*, Marseille, 2014, http://arxiv.org/abs/1409.5368.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.

[15] B. Courcelle, I. Durand, Computations by fly-automata beyond monadic second-order logic, Theoret. Comput. Sci. 619 (2016) 32–67. http://hal.archives-ouvertes.fr/hal-00828211. Short version in Proc. Conference on Algebraic Informatics, Lecture Notes in Comput. Sci. 8080 (2013) 211–222.
[16] B. Courcelle, J. Engelfriet, Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach, in: Encyclopedia of mathematics and its application, vol. 138, Cambridge University Press, 2012.
[17] B. Courcelle, P. Heggernes, D. Meister, C. Papadopoulos, U. Rotics, A characterisation of clique-width through nested partitions, Discrete Appl. Math. 187 (2015) 70–81.
[18] B. Courcelle, J. Makowsky, U. Rotics, Linear-time solvable optimization problems on graphs of bounded clique-width, Theory Comput. Syst. 33 (2000) 125–150.
[19] R. Diestel, Graph Theory, Springer, 2006.
[20] R. Downey, M. Fellows, Parameterized Complexity, Springer-Verlag, 1999.
[21] R. Downey, M. Fellows, Fundamentals of Parameterized Complexity, Springer-Verlag, 2013.
[22] I. Durand, Object enumeration, in: Proc. of 5th Europeal LISP Conference, Zadar, Croatia, May 2012, pp. 43-57.
[23] W. Espelage, F. Gurski, E. Wanke, Deciding clique-width for graphs of bounded tree-width, J. Graph Algorithms Appl. 7 (2003) 141–180.
[24] M. Fellows, F. Rosamond, U. Rotics, S. Szeider, Clique-width is NP-complete, SIAM J. Discrete Math. 23 (2009) 909–939.
[25] E. Fischer J. Makowsky, E. Ravve, Counting truth assignments of formulas of bounded tree-width or clique-width, Discrete Appl. Math. 156 (2008) 511–529.
[26] F. Fomin, S. Oum, D. Thilikos, Rank-width and tree-width of H-minor-free graphs, European J. Combin. 31 (2010) 1617–1628.
[27] A. Frank, Connectivity and networks, in: Handbook of Combinatorics, Vol.1, Elsevier, 1997, pp. 111–178.
[28] F. Gurski, E. Wanke, The tree-width of clique-width bounded graphs without $K_{n,n}$, in: Proceedings of 26th Workshop on Graphs (WG), in: Lecture Notes in Comput. Sci., 1928, 2000, pp. 196–205.
[29] P. Kolaitis, M. Vardi, Conjunctive-query containment and constraint satisfaction, J. Comput. System Sci. 61 (2000) 302–332.
[30] S. Oum, Rank-width is less than or equal to branch-width, J. Graph Theory 57 (2008) 239–244.
[31] S. Oum, Approximating rank-width and clique-width quickly, ACM Trans. Algorithms 5 (1) (2008).
[32] D. Wood, On tree-partition-width, European J. Combin. 30 (2009) 1245–1253.

Please cite this article in press as: B. Courcelle, From tree-decompositions to clique-width terms, Discrete Applied Mathematics (2017), http://dx.doi.org/10.1016/j.dam.2017.04.040.