Clique-width and tree-width of sparse graphs

Bruno Courcelle
Labri, CNRS and Bordeaux University*
33405 Talence, France
email: courcell@labri.fr

June 10, 2015

Abstract

Tree-width and clique-width are two important graph complexity measures that serve as parameters in many fixed-parameter tractable (FPT) algorithms. The same classes of sparse graphs, in particular of planar graphs and of graphs of bounded degree have bounded tree-width and bounded clique-width. We prove that, for sparse graphs, clique-width is polynomially bounded in terms of tree-width. For planar and incidence graphs, we establish linear bounds. Our motivation is the construction of FPT algorithms based on automata that take as input the algebraic terms denoting graphs of bounded clique-width. These algorithms can check properties of graphs of bounded tree-width expressed by monadic second-order formulas written with edge set quantifications. We give an algorithm that transforms tree-decompositions into clique-width terms that match the proved upper-bounds. *keywords*: tree-width; clique-width; graph decomposition; sparse graph; incidence graph

Introduction

Tree-width and clique-width are important graph complexity measures that occur as parameters in many fixed-parameter tractable (FPT) algorithms [7, 9, 11, 12, 14]. They are also important for the theory of graph structuring and for the description of graph classes by forbidden subgraphs, minors and vertex-minors. Both notions are based on certain hierarchical graph decompositions, and the associated FPT algorithms use dynamic programming on these decompositions. To be usable, they need input graphs of "small" tree-width or clique-width, that furthermore are given with the relevant decompositions.

*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. Parts of it have been presented to a worshop on graph decompositions at CIRM (Marseille, France) in January 2015.

Each class of simple graphs of bounded tree-width has bounded clique-width, hence an algorithm that is FPT for clique-width is FPT for tree-width. However, the clique-width of a graph can be exponential in its tree-width. Hence an algorithm that works well for graphs G of "small" clique-width k may not work for graphs of tree-width k because these graphs may have clique-width at least $2^{k/2}$ [2]. We investigate classes of graphs for which tree-width and clique-width are polynomially, or even better, linearly related.

Our main results

(1) We prove that tree-width and clique-width are linearly related for planar graphs, for graphs of bounded degree and for incidence graphs. We also prove that[1] $cwd(G) = O(twd(G)^q)$ if G is *uniformly q-sparse* graphs (also called q-degenerate if it is undirected) and that $cwd(Bip(H)) = O(twd(H)^{q-1})$ if H is a *q-hypergraph*, i.e., a hypergraph with hyperedges of size at most q and $Bip(H)$ is the corresponding bipartite graph. We obtain bounds for planar graphs and uniformly q-sparse graphs that improve similar bounds established for the corresponding undirected graphs in [15] by means of rank-width. Our results concern directed as well as undirected graphs. We get very short proofs by using a characterization of clique-width in terms of *partition trees* [8].

(2) We also give an algorithm that converts an *arbitrary tree-decomposition* (T, f) into a *clique-width term*[2] that witnesses the above bounds on clique-width ; the syntactic tree of the obtained clique-width term t is (up to a small transformation) the tree of the given tree-decomposition ; the computation time is $O(q^2.k.n)$ where n is the number of vertices, k is the width of the given tree-decomposition and q is the width of the produced term t.

Furthermore, the *reduced term* of t (a notion introduced in [18] for defining *relative clique-width*) is a variant T' of T. (We omit technical details). This algorithm determines the relative clique-width of G with respect to T'.

As a compact data-structure for tree-decompositions, we use *normal trees*: a rooted tree T is normal for a graph G if its nodes are the vertices of G and any two adjacent vertices of G are comparable with respect to the ancestor relation of T ; it has a *width*, that is the width of the tree-decomposition it represents.

Applications

(3) These results apply to the algorithms based on *fly-automata*[3] developped in [5, 6] and implemented in the system AUTOGRAPH. This system can check properties expressed in *monadic second-order* (*MS*) logic of graphs given by clique-width terms. It can even compute values attached to such graphs, e.g., the number of p-colorings. Hence, if a graph is given by a tree-decomposition of

[1] We denote the clique-width of a graph G by $cwd(G)$ and its tree-width by $twd(G)$.

[2] Clique-width is defined algebraically from terms that define graphs. See Definition 3.

[3] The finite automata arising from monadic second-order formulas are much too large to be implemented in the usual way by lists of transitions. To overcome this difficulty, we use *fly-automata*: their 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.

"small" width k and the associated clique-width term t has width $f(k)$ for some reasonable function f, AUTOGRAPH can be used for it, by taking t as input.

(4) Furthermore, the clique-width of the incidence graph $Inc(G)$ of a graph G of tree-width k is less that $2k+4$, and a graph property expressed by a *monadic second-order formula using edge quantifications* (an MS_2 *formula*, cf. [7, 4]) is an MS property of $Inc(G)$. This shows that AUTOGRAPH can also be used (through the transformation of a tree-decomposition of an input graph G into a clique-width term denoting $Inc(G)$) for checking such graph properties for graphs of "small" tree-width. This extension is developped in [4].

Summary of the article.

After a preliminary section devoted to a review of definitions, notation and known results, we give in Section 2 our main construction : it transforms a normal tree that encodes a tree-decomposition into a partition tree that represents a clique-width term. In Section 3, we prove that $cwd(G) = O(twd(G))$ for planar graphs and that $cwd(G) = O(twd(G)^q)$ for uniformly q-sparse graphs. In Section 4, we consider q-hypergraphs. A q-hypergraph H can be viewed as a bipartite graph $Bip(H)$ whose vertices of one side have degree at most q, hence, it is uniformly q-sparse. We improve for these bipartite graphs the construction of Section 3 and we prove that $cwd(Bip(H)) = O(twd(H)^{q-1})$. We also handle incidence graphs. In Section 5, we detail the algorithm of Section 2.

Acknowledgement : I thank I.Durand, S.Oum and M. Kanté for their useful comments.

1 Preliminaries

1.1 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.

A *partition* π of a set X is a set of nonempty pairwise disjoint sets whose union is X. These sets are called the *parts* of π. We say that π *refines partially* a partition π' of $X' \supseteq X$ if every part Y of π is included in some part Y' of π'. We denote this by $\pi \sqsubseteq_p \pi'$. It *refines* π' if $\pi \sqsubseteq_p \pi'$ and $X' = X$. The equivalence classes of an equivalence relation on X form a partition of this set. The union of two disjoint sets is denoted by $\uplus$.

If $0 < m < k$, we define $\gamma(k,m)$ as the number of subsets of $\{1,...,k\}$ of cardinality at most m. This number is $1+k+...+\binom{k}{m} < m.\binom{k}{m} < k^m/(m-1)!$ if $m > 1$, and is $1+k$ if $m = 1$.

All trees, graphs and hypergraphs are finite.

Trees

Trees are always rooted; N_T denotes the set of *nodes* of a tree T and $\leq_T$ is its *ancestor relation*, a partial order on N_T; the *root* is the unique maximal element,

the *leaves* are the minimal elements and an *inner node* is a node that is not a leaf. We denote by $p_T(u)$ the father (the closest ancestor) of a node u, by L_T the set of leaves of T, and by $L_T(u)$ the set of leaves below or equal to $u \in N_T$. We omit the subscript T if the tree T is clear from the context. 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 $T_{\geq}(u)$.

Graphs

Graphs are simple (*i.e.*, loop-free and without parallel edges), either directed or undirected. 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 $Und(G)$ with $edg_{Und(G)} := edg_G \cup edg_G^{-1}$.

If G is directed and $x \in V_G$, then $N_G^+(x)$ is the set of vertices such that $x \to_G y$ (*i.e.*, there is an edge from x to y), $N_G^-(x)$ is the set of those such that $y \to_G x$ and $N_G(x) := N_G^+(x) \cup N_G^-(x)$ is the set of *neighbours* of x. If G is undirected, then $N_G(x)$ is the set of neighbours of x. For a set 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 directed, $X \subseteq V_G, Y \subseteq X^c := V_G - X$, we define on X the equivalence relation:

$$x \sim_{X,Y} x' :\Longleftrightarrow N_G^+(x) \cap Y = N_G^+(x') \cap Y \\ \text{and } N_G^-(x) \cap Y = N_G^-(x') \cap Y.$$

We denote by $\Omega(X, Y)$ the corresponding partition of X and by $\Omega(X)$ the partition $\Omega(X, X^c)$. If G is undirected, these definitions apply with $N_G = N_G^+ = N_G^-$.

Let G be undirected and Y be a set of vertices of cardinality k. Then $|\Omega(X, Y)| \leq 2^k$. If furthermore $1 < m < k$ and $|N_G(x)| \leq m$ for all $x \in X$, then $|\Omega(X, Y)| \leq \gamma(k, m) < k^m/(m-1)!$. If G is directed, each edge of $Und(G)$ between x and y can come from three possible configurations of edges between these vertices, hence, if $|Y| = k$, we have $|\Omega(X, Y)| \leq (1+3)^k = 4^k$, and if $\left|N_{Und(G)}(x)\right| \leq m < k$ for all $x \in X$, we have $|\Omega(X, Y)| < 3^m.k^m/(m-1)!$. In both cases, We define $\lambda_G(Y) := |\Omega(Y^c)|$ and, for an integer k, we define $\overline{\lambda}_G(k)$ as the maximum value of $\lambda_G(Y)$ for a set Y of cardinality at most k. (These definitions are as in [15] where actually, only undirected graphs are considered.)

Sparse graphs

A graph G is *uniformly m-sparse* if $|E_H| \leq m.|V_H|$ for every (undirected) subgraph H of $Und(G)$. An undirected graph G is uniformly m-sparse if and only if it has an orientation of indegree at most m, *i.e.,* is m-degenerate. This fact is proved in [16] and in [7], Proposition 9.40.

Every planar graph is uniformly 3-sparse. An undirected graph is uniformly $\lceil d/2 \rceil$-sparse if its maximum degree is d because 2. $|E_G|$ is the sum of degrees of its vertices and the same holds for its subgraphs.

We denote by $\mathcal{K}_r$ the class of graphs G such that $Und(G)$ does not contain a subgraph isomorphic to the complete bipartite $K_{r,r}$. Every uniformly m-sparse graph belongs to $\mathcal{K}_{2m+1}$, but for every r and m, there are graphs in $\mathcal{K}_r$ that are not uniformly m-sparse (because there is a constant c such that, if $r \geq 3$, there is a graph having n vertices and at least $c.n^{2-2/(r+1)}$ edges, see [10], Section 7.1).

Definitions 1: *Tree-decompositions and normal trees.*

(a) The tree T of a tree-decomposition (T, f) of a graph G is always rooted; we say that (T, f) is *1-downwards increasing* ([7], Definition 2.66) if, for each node u of T, $|f(u) - f(p_T(u))| = 1$. We denote by $\overline{u}$ the unique vertex in $f(u) - f(p_T(u))$. Every tree-decomposition can be transformed into a 1-downwards increasing tree-decomposition of the same graph and that has the same width (*proof sketch*: we first contract the edges $u - p_T(u)$ of T such that $f(u) \subseteq f(p_T(u))$; if $|f'(u) - f'(p_{T'}(u))| > 1$ for some node u of the resulting tree-decomposition (T', f'), we add intermediate nodes between u and $p_{T'}(u)$, and we do this for all such nodes u). Hence, there is no loss of generality in considering only 1-downwards increasing tree-decompositions, which we will do. For such a tree-decomposition (T, f), we will always identify a node u of T and the vertex $\overline{u}$ of G, hence, we will have $N_T = V_G$. With this convention, each vertex x in $f(u)$ satisfies $u \leq_T x$. It is thus the maximal node w such that $x \in f(w)$.

(b) A tree T is *normal for a graph* G if $N_T = V_G$ and the two ends of each edge are comparable under $<_T$. A depth-first spanning tree is thus normal, but in a normal tree T, we do not require that two adjacent nodes of T are adjacent in the graph.

(c) If T is a normal tree for G and $u \in N_T$, we define

$$f_T(u) := \{u\} \cup (T_>(u) \cap N_G(T_\leq(u)))$$

i.e., $f_T(u)$ consists of u (that is also a vertex of G) and its ancestors that are adjacent to some vertex $x \leq_T u$. We define its *width* as the maximal cardinality of a set $T_>(u) \cap N_G(T_\leq(u))$. Then (T, f_T) is a 1-downwards increasing tree-decomposition of G : if $y \in f_T(u) \cap f_T(v)$, then $u, v \leq_T y$ and 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 y belongs to each set $f_T(w)$ for w on the undirected path between u and v. The other conditions are obvious to check. The width of the tree-decomposition (T, f_T) is that of T.

Lemma 2 : Let (T, f) be a 1-downwards increasing tree-decomposition of a graph G.

(1) T is normal for G; if x and y are adjacent, then $x <_T y$ and $y \in f(x)$ or $y <_T x$ and $x \in f(y)$.

(2) $f_T(u) \subseteq f(u)$ for each $u \in N_T$.

(3) If $u \in N_T$, $x \in V_G$ and $x \leq_T u$, then $N_G(x) \subseteq T_<(u) \uplus f_T(u) \subseteq T_<(u) \uplus f(u)$.

Proof: (1) Let x, y be adjacent. Then, $x, y \in f(u)$ for some $u \in N_T$ and, as observed above, $u \leq_T x$ and $u \leq_T y$. Hence, x and y are comparable under $\leq_T$. Then, either $x <_T y$ and $y \in f(x)$ because x is on the path between u and y in T or, similarly, $y <_T x$ and $x \in f(y)$.

(2) Let $y \in f_T(u)$. If $y = u$, then $y \in f(u)$. Otherwise, we have $x \leq_T u <_T y$ for some x adjacent to y. Then, $y \in f(x)$ by (1), hence, $y \in f(u)$ because $x \leq_T u <_T y$, $y \in f(y)$ and u is between x and y in T.

(3) Let $u \in N_T$, $x \in V_G$ be such that $x \leq_T u$, and $y \in N_G(x)$. If $y <_T x$, then $y \in T_<(u)$. Otherwise, $x <_T y$. If $y <_T u$, then $y \in T_<(u)$, otherwise, as u and y are comparable, $u \leq_T y$, and so $y \in f_T(u)$ by the definition of f_T. The sets $T_<(u)$ and $f(u)$ are disjoint by the fact that $x \in f(u)$ implies $u \leq_T x$. □

Figure 1 shows a graph G and the tree T of a 1-downwards increasing tree-decomposition (T, f). The function f is defined in the following table. The function f_T is as 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.

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

Normal trees offer very compact encodings of 1-downwards increasing tree-decompositions of the form (T, f_T) : the tree T can be represented by the mapping $p_T : V_G \to V_G$ or any appropriate data structure. The family of sets $f_T(u)$ for $u \in V_G$ can be computed in time $O(n(r+1))$ where $n = |V_G|$ and r is the number of edges of G not in T.

As explained in Example 5.2(4) of [7], the relational structure $\langle V_G, edg_G, son_T \rangle$ where son_T is the binary son relation of T is encodes in a compact way the graph G and a tree-decomposition (T, f_T) of it : a monadic second-order formula ψ can express that T defined by $\langle V_G, son_T \rangle$ is actually a normal tree for G defined by $\langle V_G, edg_G \rangle$ and another monadic second-order formula $\varphi(u, x)$ can express that, if ψ is true, then x belongs to $f_T(u)$.

Definition 3 : *Clique-width*

(a) The *clique-width* of a graph is defined from operations that construct graphs equipped with vertex labels. Let C be a finite set of labels. A *C-graph* is a triple $G = (V_G, edg_G, \pi_G)$ where π_G is a mapping: $V_G \to C$. We define F_C as the following set of operations on C-graphs: $\oplus$ is the union of two disjoint C-graphs, the unary operation $relab_h$ changes every vertex label a into $h(a)$ where h is a mapping from C to C, the unary operation $\overrightarrow{add}_{a,b}$, for $a \neq b$, adds

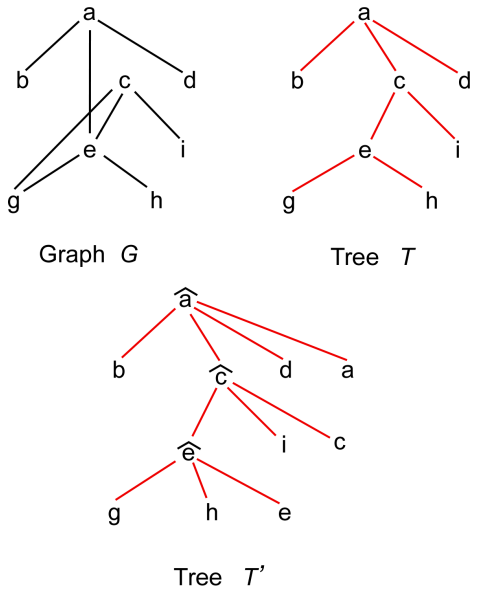

Figure 1: A graph G, a tree T and the tree T' of Proposition 7.

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 $\mathbf{a}$ denotes an isolated vertex labelled by a. For building undirected graphs, we use $add_{a,b}$ to add undirected edges. It is useful to write $\mathbf{a}(x)$ instead of $\mathbf{a}$ in order to indicate the corresponding vertex x.

Every term t in $T(F_C)$, called a *clique-width term*, denotes a C-graph $val(t)$ (for details see [7] or [5]). The clique-width of a graph G, denoted by $cwd(G)$, is the least integer k such that G is isomorphic, up to vertex labels, to $val(t)$ for some t in $T(F_{\{1,\ldots,k\}})$. Our proofs will use a characterization of clique-width not using vertex labels (see Definition 4).

(b) *Relative clique-width.* As disjoint union is associative and commutative, one can make it into an operation of variable arity and define $\oplus(t_1, ..., t_m)$ for $m \geq 2$ as $t_1 \oplus (... \oplus t_m)...)$ or with any other arrangement of parentheses, even after permuting the arguments. We denote by F_C^+ the set F_C augmented with the operations $\oplus(...)$. The notion of clique-width is not modified by using F_C^+ instead of F_C. When writing terms over F_C^+, we will denote with prefix notation the occurrences of $\oplus$ of arity 2, as those of larger arity.

Let $t \in T(F_C^+)$ denote a C-graph $val(t) = G$. Let $red(t)$ be the corresponding *reduced term*: it is obtained from t by removing the unary operations and replacing each nullary symbol $\mathbf{a}(x)$ by the corresponding vertex x of G. For example, if $t = add_{a,b}(\oplus(add_{b,c}(\oplus(\mathbf{b}(1), \mathbf{c}(2))), add_{a,c}(\oplus(\mathbf{a}(3), \mathbf{c}(4))), \mathbf{a}(5)))$, then $red(t) = \oplus(\oplus(1, 2), \oplus(3, 4), 5))$. We will also consider $red(t)$ as a rooted tree

whose leaves are the vertices.

If G is a graph and r is a reduced term whose set of nullary symbols is V_G, we denote by $rcwd(G,r)$ the minimum number of labels in a term $t \in T(F_C^+)$ such that $val(t) = G$ up to vertex labels and $red(t) = r$. This value is called the *relative clique-width of G with respect to r*. Its computation is NP-complete [19]. The present definition is a small generalization of that of [18] because we do not restrict $\oplus$ to be binary and we allow graphs to be directed.

Replacing $\oplus(t_1, ..., t_m)$ by $t_1 \oplus (... \oplus t_m))...)$ modifies the relative clique-width. Consider for an example the graph H (the "house") consisting of the cycle $1-2-3-4-5-1$ augmented with the edge $1-3$. Let $r = \oplus(2, \oplus(1,3), \oplus(4,5))$. It is not hard to see that $cwd(G,r) = 5$. If $r' = \oplus(2, \oplus(\oplus(1,3), \oplus(4,5)))$, then $cwd(G,r') = 4$.

(c) *Rank-width* is a complexity measure for undirected graphs defined and investigated in [15, 20, 21, 22]. The rank-width $rwd(G)$ of an undirected graph G satisfies the inequalities:

$$rwd(G) \leq twd(G) + 1 \text{ and } rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1.$$

We will use rank-width only through these inequalities and the results of [15]. The rank-width of directed graphs is defined and studied in [17]. It is also equivalent to clique-width.

Definition 4 : *Partition trees.*

A *partition tree* of a graph G is a pair (T, ϖ) such that T is a rooted tree whose set of leaves is V_G, all inner nodes have at least two sons, and ϖ is a mapping that associates with every node u of T a partition $\varpi(u)$ of $L_T(u)$, the set of vertices of G below u, that satisfies the following conditions, for every inner node u of T with sons $u_1, ..., u_p$:

1) *Refinement*: the partition $\varpi(u_1) \uplus ... \uplus \varpi(u_p)$ of $L_T(u)$ refines $\varpi(u)$ and

2) *Adjacency*: if $(x,y) \in edg_G$, if X and Y are parts of $\varpi(u)$ such that $x \in L_T(u_i) \cap X$, $y \in L_T(u_j) \cap Y$ and $i \neq j$, then $X \neq Y$ and $X \times Y \subseteq edg_G$.

The *width* of (T, ϖ) is the maximal cardinality of $\varpi(u)$ for some node u of T. The article [8] establishes that the clique-width of a graph is the minimal width of a partition tree of this graph. The proof transforms a partition tree (T, ϖ) of G of width p into a clique-width term t using p labels such that $T = red(t)$ and $G = val(t)$. This result is proved in [8] for undirected graphs, but its proof extends to directed graphs in a straightforward manner.

Remark: If (T, ϖ) is a partition tree of G, then for each node u of T, the partition $\varpi(u)$ refines $\Omega(L_T(u))$: consider x, x' in some part of $\varpi(u)$ and $y \in V_G - L_T(u)$ such that $(x,y) \in edg_G$ or $(y,x) \in edg_G$. The adjacency

condition applied to x, y and the least common ancestor of u and y shows that $(x', y) \in edg_G$, or, respectively, $(y, x') \in edg_G$. Hence, x, x' are in a same part of $\Omega(L_T(u))$.

1.2 Tree-width compared to clique-width : a review of known results

For every graph G, we have $twd(Und(G)) = twd(G)$ and $cwd(Und(G)) \leq cwd(G)$.

Theorem 5 : For all graphs G, the following hold:
(1) $twd(G)$ is unbounded in terms of $cwd(G)$,
(2) $cwd(G) \leq 3.2^{twd(G)-1}$ 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, this proves (1); assertions (2) and (4) are proved in [2]. For assertion (3) see [7], Proposition 2.114. □

For sparse graphs, we have the following results. ($\mathcal{K}_r$ is the class of graphs G such that $Und(G)$ does not contain a subgraph isomorphic to $K_{r,r}$).

Theorem 6 : For all graphs G, the following hold:
(1) If G has maximal degree at most d (with $d \geq 1$) we have:

(1.1) $twd(G) \leq 3.d.cwd(G) - 1$,

(1.2) $cwd(G) \leq 20.d.(twd(G) + 1) + 2$.

(2) If $r \geq 2$ and $G \in \mathcal{K}_r$ we have $twd(G) \leq 3.(r-1).cwd(G) - 1$.

Proof : Assertion (2) is from [13] (also [7], Proposition 2.115) and it yields (1.1). Assertion (1.2) is proved in [3] by means of a strong result of [23]. □

Theorem 6(2) shows that, for each m, $twd(G) = O(cwd(G))$ if G is uniformly m-sparse. An opposite (polynomial) bounding will be established in Theorem 12.

2 From tree-decompositions to clique-width terms

Here is our main construction. We will make it into an algorithm in Section 5. A variant of it is in the proof of Theorem 15.

Proposition 7: Let (T, f) be the 1-downwards-increasing tree-decomposition of a graph G. If $|\Omega(T_<(u), f(u))| \leq m$ for every node u of T, then $cwd(G) \leq m+1$.

Proof: We assume that G has at least 2 vertices, so that the root of T is an inner node. The tree T is normal for G; we can assume that $f = f_T$, but this is not necessary for the following construction.

We define a partition tree (T', ϖ) of G. We first define T' :

i) for each inner node u of T, we define a new node $\widehat{u}$ ("new" means different from all nodes of T),

ii) $N_{T'} := N_T \cup \{\widehat{u} \mid u \text{ is an inner node of } T\}$,

iii) the root of T' is $\widehat{root_T}$, $p_{T'}(\widehat{u}) := \widehat{p_T(u)}$ if u is an inner node that is not $root_T$, $p_{T'}(u) := \widehat{u}$ if u is an inner node and $p_{T'}(u) := \widehat{p_T(u)}$ if u is a leaf of T.

Hence T' is a rooted tree whose leaves are the nodes of T. See Figure 1 in the previous section for an example. We now define ϖ:

iv) $\varpi(u) := \{\{u\}\}$ if u is a leaf of T' (a node of T and also a vertex of G),

v) $\varpi(\widehat{u}) := \{\{u\}\} \cup \Omega(T_<(u), f(u))$ otherwise (u is an inner node of T and also a vertex of G).

The set $L_{T'}(\widehat{u})$ of leaves of T' below $\widehat{u}$ is $T_\leq(u)$ and $\varpi(\widehat{u})$ is a partition of this set.

First, we note that $\Omega(T_<(u), f(u)) = \Omega(T_<(u))$ by Lemma 2 because the neighbours of $x \in T_<(u)$ not in this set are in $f(u)$. We check the Refinement Condition of Definition 4. Let $\widehat{u}$ be an inner node of T' with sons $w_1, ..., w_p$ and u, where each w_i is a leaf or an inner node $\widehat{u_i}$. Let x, y be distinct elements of some $X \in \varpi(\widehat{u_i})$. We verify that they belong to some $Y \in \varpi(\widehat{u})$. The set X is a part of some partition $\Omega(T_<(u_i), f(u_i))$ that is equal to $\Omega(T_<(u_i))$. Since $T_<(u_i) \subseteq T_<(u)$, we have $\Omega(T_<(u_i)) \sqsubseteq_p \Omega(T_<(u))$, hence x, y belong to some $Y \in \Omega(T_<(u)) = \Omega(T_<(u), f(u)) \subseteq \varpi(\widehat{u})$.

We now check the Adjacency Condition. Let $\widehat{u}$ be an inner node of T' with sons $w_1, ..., w_p$ as above and $w_{p+1} = u$. Consider $(x, y) \in edg_G$ and X,$Y \in \varpi(\widehat{u})$ such that $x \in X \cap L_{T'}(w_i)$, $y \in Y \cap L_{T'}(w_j)$ and $i \neq j$. As there are no edges between $L_{T'}(w_i)$ and $L_{T'}(w_j)$ if $i, j \leq p$, we must have i or j, say i, equal to $p+1$. Hence, $x = u$ and $X = \{x\} \neq Y$. If $y' \in Y$, then $y' \sim_{T_<(u), f(u)} y$, hence $(x, y') \in edg_G$ as $x \in f(u)$. Hence, $\{x\} \times Y \subseteq edg_G$ as was to be proved.

The number of classes of $\varpi(\widehat{u})$ is at most $m+1$, which gives the result by the main theorem of [8].□

Remarks 8: (1) The tree T' is $NewTree(root_T)$ where $NewTree$ is defined recursively as follows:

if u is a leaf of T, then $NewTree(u) := u$,

if $u_1, ..., u_r$ are the sons of u, then

$NewTree(u) := \widehat{u}(u, NewTree(u_1), ..., NewTree(u_r))$.

In the example of Figure 1, $T = \widehat{a}(a, b, \widehat{c}(c, \widehat{e}(e, g, h), i), d)$.

(2) The tree T' is $red(t)$ where t is the clique-width term constructed from (T', ϖ) to denote G by the method of of [8].

(3) For undirected graphs G, the article [15] uses the inequality $cwd(G) \leq 2.\overline{\lambda}_G(rwd(G)) - 1$, that gives $cwd(G) \leq 2.\overline{\lambda}_G(twd(G) + 1) - 1$. Proposition 7 yields better bounds, based on the inequality $cwd(G) \leq \overline{\lambda}_G(twd(G) + 1) + 1$ that is valid for directed graphs as well, and on some combinatorial lemmas.

3 Sparse graphs

We first consider the class $\mathcal{P}$ of (simple) planar graphs. They are uniformly 3-sparse.

3.1 Planar graphs

We know from [15] that tree-width, clique-width and rank-width are linearly equivalent for undirected graphs embeddable in a fixed surface. We establish bounds relating tree-width and clique-width for directed and undirected planar graphs, based on Proposition 7. An algorithm that constructs clique-width terms from tree-decompositions is given in Section 5. It does not use any plane embedding of the input graph.

Smoothing a vertex x of degree 2 that has neighbours y and z in an undirected graph means replacing x and its two incident edges by a single edge between y and z, and fusing the parallel edges that may result. This transformation preserves planarity.

Lemma 9: Let $G \in \mathcal{P}$ and $k \geq 3$.
1) If G is undirected, then $\overline{\lambda}_G(k) \leq 6k - 9$.
2) If G is directed, then $\overline{\lambda}_G(k) \leq 32k - 57$.
If $|Y| \leq 2$, the upper bounds are respectively 4 and 13.

Proof: 1) This assertion is Proposition 11 of [15]. We prove it for completeness and in order to prove the assertion about directed graphs. We consider disjoint sets X and Y with Y of cardinality k, and we bound the number of parts of $\Omega(X, Y)$, i.e. of classes of $\sim_{X,Y}$.

We define the *type* of a vertex of X as the number of its neighbours in Y and the *type* of an equivalence class of $\sim_{X,Y}$ as the common type of all its elements. We will prove the results for bipartite 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 $\sim_{X,Y}$ and $\Omega(X, Y)$.

The equivalence $\sim_{X,Y}$ has at most k classes of type 1 and one class of type 0, without needing the assumption that G is planar. Next we consider the vertices of type 2. We remove from G the vertices of X of type different from 2. We get a planar graph G' whose vertices that are in X have degree 2. By smoothing these vertices we get a graph $H \in \mathcal{P}$ with vertex set Y of cardinality k. Each edge of H corresponds to an equivalence class of $\sim_{X,Y}$ of type 2. Hence, there are at most $3k-6$ classes of type 2.

We now consider the vertices of type at least 3. We remove from G the vertices of X of type at most 2. We get a bipartite graph $K \in \mathcal{P}$ with edges between $V_K \cap X$ and Y. Let $a := |V_K \cap X|$. As each vertex in $V_K \cap X$ has degree at least 3 in K, $3a \leq |E_K|$. As K is planar and bipartite, $|E_K| \leq 2\,|V_K| - 4$. Hence, $3a \leq |E_K| \leq 2(a+k)-4$ which gives $a \leq 2k-4$. The number equivalence classes of type at least 3 is at most $|V_K \cap X| = a$. (Each equivalence class of type at least 3 has at most 2 elements, because otherwise, $K_{3,3}$ would be a subgraph of G that is assumed planar). Hence, we get at most $1+k+3k-6+2k-4 = 6k-9$ classes.

2) Assume now that G is directed. We consider the simple undirected graph $Und(G)$ obtained from G by forgeting edge directions and fusing any two parallel edges. It has at most one class of type 0 and at most $3k$ classes of type 1. Each class of $Und(G)$ of type 2 splits into at most 9 classes of G with same neighbour vertices, which gives at most $9(3k-6)$ classes. The above proof for an undirected graph shows that $2k-4$ bounds the number of vertices of K of degree at least 3. This number bounds also the number equivalence classes of type at least 3 of G. Hence, we get the upper bound: $1 + 3k + 9(3k-6) + 2k - 4 = 32k - 57$.

The bounds $3k-6$ (resp. $2k-4$) on numbers of edges of simple planar graphs (resp. 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 10 : 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 9 and Proposition 7, 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 and $6(k+1) - 9 + 1 = 6k-2$ for an undirected graph. □

The article [15] proves that, for G undirected and 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 9(1) are replaced by $3k-6+3r$ and $2k-4+2r$ respectively. The corresponding modifications of Lemma 9(2) and Theorem 10 give the bounds $cwd(G) \leq 32.twd(G) + O(r)$ for G directed and $cwd(G) \leq 6.twd(G) + O(r)$ for G undirected and embedded on some surface of genus r.

3.2 Uniformly m-sparse graphs

Lemma 11: Let $k \geq m > 1$ and G be uniformly m-sparse.
1) If G is undirected, then $\overline{\lambda}_G(k) < k.m + k^m/(m-1)!$.
2) If G is directed, then $\overline{\lambda}_G(k) < k.m + (3k)^m/(m-1)!$.

Proof: Let $k \geq m > 1$ and G be uniformly m-sparse. We let X and Y be as in the proof of Lemma 9 and we bound $|\Omega(X,Y)|$.

1) Let G be undirected and H be an orientation of G of indegree at most m. As in the proof of Lemma 9, we can assume that G is 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 m and $N_H^+(x) \subseteq Y$, $|X_1| \leq m.k$ and hence, $|\Omega(X_1,Y)| \leq |X_1| \leq m.k$. ($\Omega$ 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 m. There are at most $\gamma(k,m) < k^m/(m-1)!$ such sets by an observation in Section 1.1, hence, $|\Omega(X_2,Y)| < k^m/(m-1)!$. 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 m-sparse. We still have $|\Omega(X_1,Y)| \leq m.k$ but $|\Omega(X_2,Y)| < 3^m.k^m/(m-1)!$ as each edge of H can arise from three configurations of edges in G.□

Theorem 12: For each m, if G is uniformly m-sparse, then $cwd(G) = O(twd(G)^m)$.

Proof: Immediate consequence of Lemma 11 and Proposition 7. □

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 Theorem 6(1.2) gives a (better) linear upper-bound. Since planar graphs are uniformly 3-sparse, we get $cwd(G) = O(twd(G)^3)$ for them, but Theorem 10 also gives linear upper-bounds.

The undirected graphs that omit a fixed graph H as a minor or as a topological minor are uniformly m-sparse for some m. For these classes, tree-width, clique-width and rank-width are linearly equivalent by Theorems 10 and 16 of [15].

Every uniformly m-sparse graph G belongs to $\mathcal{K}_{2m+1}$. For undirected graphs H in $\mathcal{K}_r$, we have $cwd(H) = O(\gamma(twd(H),r)) = O(twd(H)^r)$ by a result of [15], which yields $cwd(G) = O(twd(G)^{2m+1})$ but the bound of Theorem 12 is better.

4 Bipartite graphs and hypergraphs

A bipartite graph G is *d-bounded* if all vertices of one part of V_G have degree at most d. For such a graph, $Und(G)$ has an orientation of indegree at most d, hence Theorem 12 gives $cwd(G) = O(twd(G)^d)$.

Bipartite gaphs are interesting for many reasons. In particular, they can encode incidence graphs and hypergraphs as we will see, and also SAT problems [14].

Definition 13 : *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 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 2 vertices and each vertex belongs to some hyperedge (similarly, graphs are loop-free without isolated vertices). 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 except that every 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 notion of a *1-downwards increasing* decomposition are as for $K(H)$.

Figure 2 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 width 5; the function f is defined in the following table.

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

Lemma 14 : (1) For every hypergraph H, $twd(Bip(H)) \leq twd(H) + 1$ and $twd(H)$ is unbounded in terms of $twd(Bip(H))$.

(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 width $twd(Bip(H)) + 1$.

The reader will find an example such that $twd(Bip(H)) = 3$ and $twd(H) = 2$. For each n, we have $twd(Bip(H)) = 1, twd(H) = n$ if H has one hyperedge with $n + 1$ vertices.

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

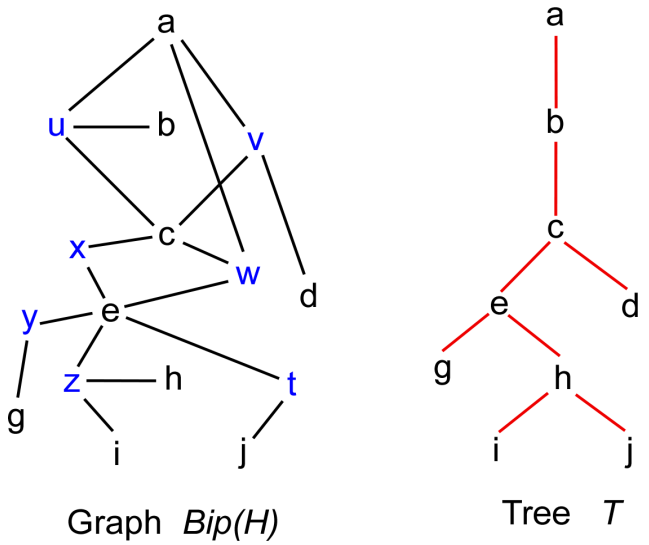

Figure 2: $Bip(H)$ and the tree T of a tree-decomposition of H.

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

Then (T, f') is a tree-decomposition of H and $|f'(u)| \leq q.|f(u)| \leq q(k+1)$ which yields the result.□

Hence, the tree-width of a q-hypergraph and that of its associated bipartite graph are linearly related. If H is a q-hypergraph, we have $cwd(Bip(H)) = O(twd(Bip(H))^q)$ by Theorem 12. We now relate the clique-width of $Bip(H)$ to the tree-width of H. We have $cwd(Bip(H)) = O(twd(H)^q)$ by this fact and Lemma 14 but we can do better[4].

Theorem 15 : For every q-hypergraph H, we have $cwd(Bip(H)) = O(twd(H)^{q-1})$.

Proof: We modify the construction of Proposition 7. Let H be a q-hypergraph and $G := Bip(H)$ where $V_G = V_H \cup E_H$. Let (T, f) be a 1-downwards increasing tree-decomposition of H (equivalently, of $K(H)$), with $N_T = V_H$. Hence, T is normal for $K(H)$ and each vertex x is the maximal node u of T such that $x \in f(u)$. Every hyperedge e has all its vertices in some set $f(w)$. Hence $w \leq_T x$ for each vertex x of e. These vertices are on the path in T between the root and w, and we call the smallest one with respect to $\leq_T$ the *first vertex* of e. If $X \subseteq V_H$ we denote by $E(X)$ the set of hyperedges having at least one vertex in X.

[4]A similar result in [1] 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.

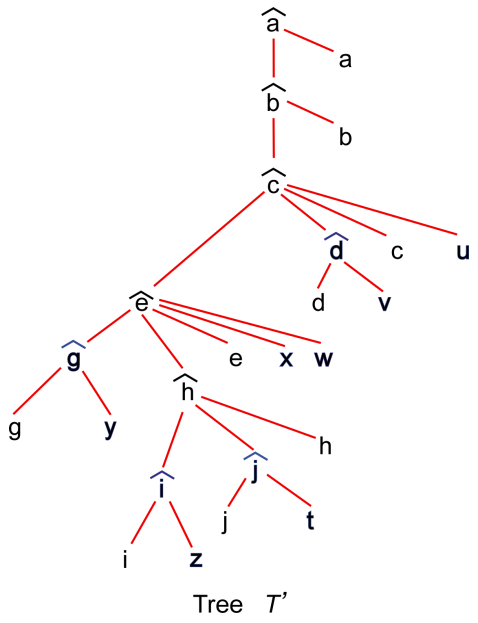

Figure 3: Tree T' for the example of Figure 2.

As in the proof of Proposition 7, we construct from (T, f) a partition tree (T', ϖ) of G. We first define T' :

i) for each $u \in N_T$, we define a new node $\widehat{u}$,

ii) $N_{T'} := N_T \cup E_H \cup \{\widehat{u} \mid u \in N_T\}$,

iii) the root of T' is $\widehat{root_T}$ and if u is not the root, then $p_{T'}(\widehat{u}) := \widehat{p_T(u)}$,

iv) if $u \in N_T$, then $p_{T'}(u) := \widehat{u}$,

v) if $e \in E_H$ and its first vertex is u, then $p_{T'}(e) := \widehat{u}$.

Hence $L_{T'} = N_T \cup E_H$ and $L_{T'}(\widehat{u}) = T_{\leq}(u) \cup E(T_{\leq}(u))$ if u is an inner node.

The tree T' is $NewTree(root_T)$ where $NewTree$ is defined recursively, similarly as in Remark 8(1), by :

$NewTree(u) := \widehat{u}(u, NewTree(u_1), ..., NewTree(u_r), e_1, ..., e_s)$,

if $u_1, ..., u_r$ are the sons of u and $e_1, ..., e_s$ are the hyperedges whose first vertex is u.

Figure 3 shows T' for H and T of Figure 2. The first vertices of hyperedges u, v, w, x are respectively c, d, e and e. The left-right ordering of sons of a node is irrelevant.

We now define ϖ, where the mapping Ω is relative to the graph $G = Bip(H)$:

vi) $\varpi(u) := \{\{u\}\}$ if $u \in N_T \cup E_H$,

vii) $\varpi(\widehat{u}) := \{\{u\}\} \cup \{T_<(u)\} \cup \Omega(E(T_\leq(u)), f(u))$.

In order to verify the two conditions of Definition 4, we make two observations. First, if $x \in T_\leq(u)$ and $e \in E(\{x\}) = N_G(x)$, then $N_G(e) \subseteq N_{K(H)}(x) \subseteq T_<(u) \uplus f_T(u) \subseteq T_<(u) \uplus f(u)$ by Lemma 2. Second, for each e with first vertex u, $N_G(e) \cap T_<(u) = \emptyset$, hence, $N_G(e) \subseteq f(u)$.

We check the Refinement Condition. The only nontrivial case to check is $\varpi(\widehat{u}) \sqsubseteq_p \varpi(\widehat{u'})$ where $u' = p_T(u)$:

$\varpi(\widehat{u}) := \{\{u\}\} \cup \{T_<(u))\} \cup \Omega(E(T_\leq(u)), f(u))$,

$\varpi(\widehat{u'}) := \{\{u'\}\} \cup \{T_<(u')\} \cup \Omega(E(T_\leq(u')), f(u'))$.

We have $T_<(u) \subseteq T_<(u')$ and $E(T_\leq(u)) \subseteq E(T_\leq(u'))$.

Every neighbour in $f(u')$ of a hyperedge in $Z := E(T_\leq(u))$ belongs to $f(u)$ by the second observation made above. Hence, if two hyperedges of Z are in a same part of $\Omega(Z, f(u))$, they are in the same part of $\Omega(E(T_\leq(u')), f(u'))$.

We now check the Adjacency Condition. Let $(x, e) \in (X \times Y) \cap edg_G \subseteq V_H \times E_H$ where X, Y are parts of $\varpi(\widehat{u})$. The sons of $\widehat{u}$ in T' are u, the hyperedges $e_1, ..., e_p$ having first vertex u and the nodes $\widehat{u_1}, ..., \widehat{u_r}$ such that $u_1, ..., u_r$ are the sons of u in T. We also assume that x and e are below different sons of $\widehat{u}$. Clearly, $X \neq Y$.

Case 1: $x = u$ and $e \in Y \subseteq E(T_\leq(u))$. Then $X \times Y = \{u\} \times Y \subseteq inc_H = edg_G$ because Y is a part of $\Omega(E(T_\leq(u)), f(u))$ and $u \in f(u)$.

Case 2: $x \in T_<(u)$ and $e \in Y \subseteq E(T_\leq(u))$. Then, $x \in T_\leq(u_i)$ for some i and the first vertex of e is in $T_\leq(u_i)$. This contradicts the hypothesis that x and e are below different sons of $\widehat{u}$. Hence, this case cannot happen.

The width of the partition tree (T', ϖ) is the maximum cardinality of a set $\varpi(\widehat{u}) = \{\{u\}\} \cup \{T_<(u)\} \cup \Omega(E(T_\leq(u)), f(u))$. We have $E(T_\leq(u)) = E(T_<(u)) \cup \{e_1, ..., e_p\}$ where $e_1, ..., e_p$ are the hyperedges with first vertex u. If the width of (T, f) is k, then $|f(u)| \leq k + 1$. Each hyperedge $e \in E_H$ has at most q neighbours. We first consider the hyperedges in $E(T_<(u))$ that contain u. They have at most $q - 1$ other vertices in the set $f(u) - \{u\}$ of cardinality at most k. Hence, $\Omega(E(T_<(u)) \cup \{e_1, ..., e_p\}, f(u))$ has at most $\gamma(k, q - 1)$ parts consisting of such hyperedges (among whose are $e_1, ..., e_p$). Its parts that do not contain u contain only hyperedges e having at least one neighbour in $T_<(u)$. So these hyperedges have at most $q-1$ vertices in $f(u)$ and there are at most $\gamma(k+1, q-1)$ such parts. This gives $|\varpi(\widehat{u})| \leq 2 + \gamma(k, q - 1) + \gamma(k + 1, q - 1) = O(k^{q-1})$.□

Incidence graphs

Let G be undirected. Its incidence graph $Inc(G)$ is the bipartite graph $(V_G \cup E_G, inc_G)$ such that $inc_G := \{(v, e) \in V_G \times E_G \mid v \text{ is a vertex of } e\}$. 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 $inc_G := \{(v,e) \in V_G \times E_G \mid e : v \to_G w \text{ for some vertex } w\} \cup \{(e,v) \in E_G \times V_G \mid e : w \to_G v \text{ for some vertex } w\}$.

Tree-width and clique-width for G and $Inc(G)$ are related as follows:

$twd(G) \leq twd(Inc(G)) \leq twd(G) + 1$ and[5]

$twd(Inc(G)) \leq 6.cwd(Inc(G)) - 1$ by Theorem 6(2).

The proof of Theorem 15 yields $cwd(Inc(G)) \leq 2.twd(G) + 5$ for G undirected. However, a slightly more complicated proof gives the following bounds proved in [1] in a different way.

Theorem 16: We have $cwd(Inc(G)) \leq twd(G) + 3$ if G is undirected and $cwd(Inc(G)) \leq 2.twd(G) + 4$ if it is directed.

Proof sketch : Let (T, f) be a 1-downwards increasing tree-decomposition of width k of G, undirected. We define a tree T' with set of nodes $V_G \cup E_G \cup \{\widehat{u} \mid u \in V_G \cup E_G\}$ by $T' := NT(root_T)$ such that (cf. Remark 8(1) and Theorem 15), for $u \in N_T$:

$NT(u) := \widehat{e_s}(e_s, \widehat{e_{s-1}}(e_{s-1}, ..., \widehat{e_1}(e_1, \widehat{u}(u, NT(u_1), ..., NT(u_r))...))$,
where $u_1, ..., u_r$ are the sons of u and $e_1, ..., e_s$ are the edges whose first vertex is u.

The definition of ϖ will use the following notation for $u \in N_T = V_G$:

$E_<(u)$ is the set of edges of G whose two ends are in $T_<(u)$,

$E_\leq(u)$ is the set of edges with one end in $T_<(u)$, the other being u,

$E_{<>}(u)$ is the set of edges with one end in $T_<(u)$, the other being an ancestor of u.

For a leaf x of T' ($x \in V_G \cup E_G$),

$\varpi(x) := \{\{x\}\}$.

For an inner node u of T:

$\varpi(\widehat{u}) := \{\{u\}, E_\leq(u), E_<(u) \cup T_<(u)\} \cup \Omega(E_{<>}(u), f(u) - \{u\})$.

For an edge e_i as in the definition of NT :

$\varpi(\widehat{e_i}) := \{\{u\}, \{e_i\}, E_\leq(u) \cup E_<(u) \cup T_<(u)\} \cup$
$\Omega(E_{<>}(u) \cup \{e_i, ..., e_s\}, f(u) - \{u\})$.

[5] If G is simple and undirected, then $twd(G) = twd(Inc(G))$. If G consists of two opposite directed edges, then $twd(G) = 1$ and $twd(Inc(G)) = 2$.

Claim 1 : (T', ϖ) is a partition tree of G.

Proof: Most verifications are straighforward. The least obvious one is that $\varpi(\widehat{e_i}) \sqsubseteq_p \varpi(\widehat{w})$ where $w = p_T(u)$. For that purpose, we prove :

$$\Omega(E_{<>}(u) \cup \{e_1, ..., e_s\}, f(u) - \{u\}) \sqsubseteq_p \\ \{E_{\leq}(w)\} \cup \Omega(E_{<>}(w), f(w) - \{w\}).$$

Let e, e' be in some part X of $\Omega(E_{<>}(u) \cup \{e_1, ..., e_s\}, f(u) - \{u\})$. The first vertices of e, e' are in $T_{\leq}(u)$. Their second vertices are in $f(u) - \{u\}$, hence must be equal, say to some w'. If $w' = w$, then $e, e' \in E_{\leq}(w)$. Otherwise, $w <_T w'$, $e, e' \in E_{<>}(w)$ and $w' \in f(w) - \{w\}$. Hence, e, e' are in a same part of $\Omega(E_{<>}(w), f(w) - \{w\})$, as was to be proved.□

Claim 2 : The width of (T', ϖ) is at most $k + 3$.

Proof: Consider $\varpi(\widehat{u})$. As $f(u) - \{u\}$ has at most k elements and each edge in $E_{<>}(u)$ has a unique vertex in $f(u) - \{u\}$, the partition $\Omega(E_{<>}(u), f(u) - \{u\})$ has at most k parts. This argument gives the same bound for $\varpi(\widehat{e_i})$.□

If G is directed, the construction is the same except that in the definition of $\varpi(\widehat{u})$, the set $E_{\leq}(u)$ is replaced by the two sets $E^+_{\leq}(u)$ and $E^-_{\leq}(u)$ defined respectively as the set of edges with tail in $T_<(u)$ and head u, and, vice versa, head in $T_<(u)$ and tail u. Because edges have two possible directions but there are no pairs of opposite edges, the sets $\Omega(E_{<>}(u), f(u) - \{u\})$ have at most $2k$ parts. This gives the bound $2k + 4$ for the width of (T', ϖ).□

Applications to FPT algorithms.

The system AUTOGRAPH, cf. [5, 6] checks *monadic second-order* (*MS*) properties of graphs of clique-width at most k for "small" values of k. It can even compute values associated with graphs such as their numbers of p-colorings. Input graphs must be given by clique-width terms. In order to use AUTOGRAPH for graphs G given by tree-decompositions (T, f) of width at most w, we need that the bound $g(w)$ on the clique-width of G remains in the range of acceptable values, typically is not 2^w.

Theorem 16 indicates that AUTOGRAPH can be used to check MS properties of incidence graphs $Inc(G)$ for graphs G of "small" tree-width because there is no exponential jump as in Theorem 5(2,3). Furthermore, these properties can be seen as expressed by MS formulas using *edge set quantifications*: they are more expressive than the MS formulas interpreted over (V_G, edg_G) (but bounded tree-width is necessary for having FPT algorithms). This observation is developped in [4].

5 Constructing clique-width terms from tree-decompositions

In order to use AUTOGRAPH for graphs given by tree-decompositions, we need to transform efficiently these decompositions into "good" clique-width terms.

In all our proofs that bound clique-width in terms of tree-width, a tree T' constructed from the tree T of a given tree-decomposition (see Proposition 7 and Theorem 15) is used as syntactic tree of the constructed clique-width term, except for Assertion (1.2) in Theorem 6, because its proof is based on a difficult result of [23] that restructures in a complicated way the tree T.

We present an algorithm that implements the construction of Proposition 7. From a normal tree T (the tree of a 1-downwards increasing tree-decomposition) of the given graph G, a first algorithm computes the *graph representation* $D(T', \varpi)$ of the partition tree (T', ϖ) of Proposition 7. Then, the algorithm of Proposition[6] 3.7 of [8] builds a clique-width term t of same width as ϖ such that $val(t) = G$ and $red(t) = T'$. We obtain a better computation time than the one arising from the algorithm of [18] that approximates the relative clique-width.

Definition 17 : *Graph representation of a partition tree.*

Let (T, ϖ) be a partition tree of width p of a graph G. We use the notation of Definition 4. We define a directed graph $D = D(T, \varpi)$ as follows :

$V_D := N_T \uplus W_D$ where $W_D := \{(u, X) \mid u \in N_T, X \in \varpi(u)\}$,

E_D consists of the edges of T and the following ones:

(i) the edges $u \to (u, X)$ such that $u \in N_T, X \in \varpi(u)$,

(ii) the edges $(p_T(u), X) \to (u, Y)$ such that $Y \subseteq X$,

(iii) the edges $(u, X) \to (u, Y)$ such that (X, Y) is as in the Adjacency Condition of Definition 4.

The edges of type (iii) are not present in the definition given in [8], but they make it possible to construct the clique-width term without using the adjacency matrix of G. This construction is possible in time $O(p.|E_D|)$ and yields a term of width p.

Let n be the number of vertices of G and $m := |edg_G|$ (if G is undirected, an edge counts as two directed (opposite) edges).

Claim 1 : The number of edges of D of type (iii) is at most m.

Proof : For each triple (u, X, Y) as in the Adjacency Condition, we choose an edge $x \to y$ such that $x \in X$ and $y \in Y$ that we denote by $\overline{(u, X, Y)}$. Clearly, u is the least common ancestor of x and y in T. If $u \neq u'$, then $\overline{(u, X, Y)} \neq \overline{(u', X', Y')}$. If $\overline{(u, X, Y)} = \overline{(u, X', Y')}$, we have $(X, Y) = (X', Y')$ because X, Y, X' and Y' are parts of the partition $\varpi(u)$. This proves that the number of such triples (u, X, Y) is at most m.□

Claim 2: D has $O(n.p)$ vertices and $O(m + n.p)$ edges.

[6] This proposition indicates the time bound $O(p.d + n^2)$ where d is the number of vertices of D, n the number of vertices of G. The term n^2 can be replaced by m, the number of edges of G. We get the time bound $O(p^2.n + m)$.

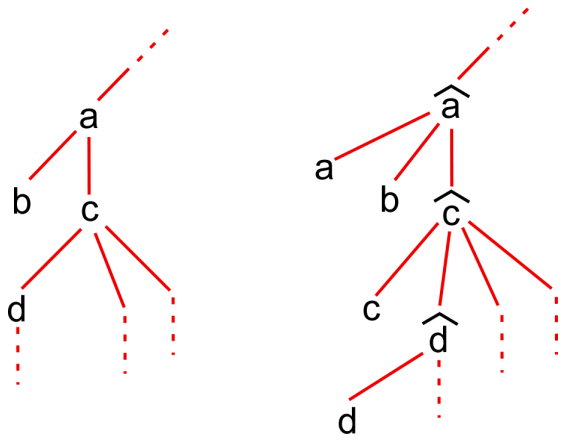

Figure 4: A tree T and the corresponding tree T'.

Proof : We have $|N_T| \leq 2n-1$ and $|W_D| \leq n + p(n-1)$. We have $|E_T| \leq 2n-2$, and D has at most $n+p(n-1)$ edges of type (i), at most $p.|E_T|$ edges of type (ii) and at most m edges of type (iii) by Claim 1. □

Algorithm 18 :*From a normal tree to the graph of a partition tree.*

We will construct $D(T', \varpi)$ for a graph G and a normal tree T for this graph, where T' is defined in the proof of Proposition 7. We recall that the set of nodes of T' is $N_{T'} = N_T \cup \{\widehat{u} \mid u \text{ is an inner node of } T\}$.

Preliminary observations.

Before presenting the algorithm, we show how the graph $D(T', \varpi)$ looks like. Figure 4 shows a portion of a tree T and the corresponding portion of T'. Figure 5 shows vertices (u, X) of $D = D(T', \varpi)$ corresponding to the nodes u of T' shown in Figure 4. Figure 5 is based of the following assumptions : $\Omega(T_<(a)) = \{A_1, A_2\}, \Omega(T_<(c)) = \{C_1, C_2\}$, $A_1 = \{b, c\} \cup C_2$, $A_2 = C_1$, and we have : $(\{a\} \times A_2) \cup (A_1 \times \{a\}) \cup (\{c\} \times C_1) \cup (\{c\} \times C_2) \subseteq edg_G$.

The Adjacency Condition produces the edges of D of type (iii): $(\widehat{a}, \{a\}) \to (\widehat{a}, A_2)$, $(\widehat{a}, A_1) \to (\widehat{a}, \{a\})$, $(\widehat{c}, \{c\}) \to (\widehat{c}, C_1)$ and $(\widehat{c}, \{c\}) \to (\widehat{a}, A_2)$. Since $b, c \in A_1$, we have edges $b \to a$ and $b \to c$ in G.

The algorithm takes as input an undirected graph G and a normal tree T for it. For each $x \in N_T = V_G$ we define :

$d(x)$ as its depth in T (the root has depth 0),

$N_i(x) := \{y \in V_G \mid x <_T y, (x, y) \in edg_G, d(y) \leq i\}$, where $i < d(x)$,

$\partial := \max\{|N_{d(x)-1}(x)| \mid x \in N_T\} = \max\{|N_i(x)| \mid x \in N_T, i < d(x)\}$.

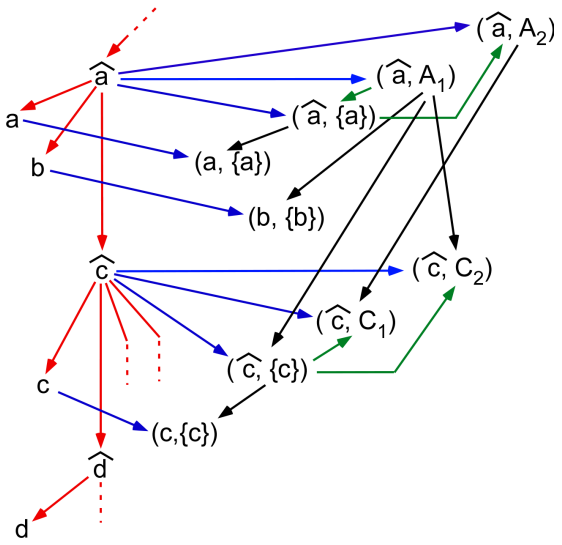

Figure 5: The graph $D(T', \varpi)$.

Fact 1 : If T is the tree of a 1-downwards increasing tree-decomposition of width k, then $\partial \leq k+1$ by Lemma 2.

Fact 2 : If $x, y <_T u$, then x and y are in a same part of $\Omega(T_<(u)) \subseteq \varpi(\widehat{u})$ if and only if $N_{d(u)}(x) = N_{d(u)}(y)$.

If $X \in \Omega(T_<(u))$, we can define $N_{d(u)}(X)$ as $N_{d(u)}(x)$ for any x in X because this set is the same for all such x.

Fact 3 : Let $X \in \Omega(T_<(u))$ and $Y \in \Omega(T_<(u'))$ where u'' is the common father of u and u' (we may have $u = u'$). These sets are included in a same part Z of $\Omega(T_<(u''))$ if and only if $N_{d(u)}(X) - \{u\} = N_{d(u')}(Y) - \{u'\}$, which is easy to decide, and then $N_{d(u'')}(Z) = N_{d(u)}(X) - \{u\}$.

We now review the graph $D(T', \varpi)$. Its vertices are the nodes of T', the pairs $(u, \{u\})$ for all $u \in N_T = V_G$, and the pairs $(\widehat{u}, \{u\})$ and $(\widehat{u}, X)$ for all inner nodes u of T and parts X of $\Omega(T_<(u))$. Its edges are those of T' and the following ones:

of type (i) : $u \to (u, \{u\})$, $\widehat{u} \to (\widehat{u}, \{u\})$ and
$\widehat{u} \to (\widehat{u}, X)$ for $X \in \Omega(T_<(u))$,
of type (ii) : $(\widehat{u}, \{u\}) \to (u, \{u\})$ if u is an inner node, and, letting u' be the father of u:
$(\widehat{u'}, X) \to (u, \{u\})$ if u is a leaf and $u \in X$,

$(\widehat{u'}, X) \to (\widehat{u}, \{u\})$ if u is an inner node and $u \in X$,

$(\widehat{u'}, X) \to (\widehat{u}, Y)$ if $Y \subseteq X$.

of type (iii) : $(\widehat{u}, X) \to (\widehat{u}, \{u\})$ and $(\widehat{u}, \{u\}) \to (\widehat{u}, X)$ if $u \in X$.

Since G is undirected, we have pairs of opposite edges of type (iii). By the definition of (T', ϖ), there are no edges of the form $(\widehat{u}, X) \to (\widehat{u}, Y)$ because there is no edge between two vertices in $T_<(u)$.

The algorithm for G undirected.

Its input is T together with, for each $x \in N_T$, the set $N_{d(x)-1}(x)$. (These sets are easily computed from standard data structures for G and T). The graph G, assumed without isolated vertices, is completely defined from this data, whose size is $O(|E_G|)$.

We assume that T' has already been constructed. By traversing it (or rather T) in a bottom-up way, we build the directed graph $D(T', \varpi)$ by adding successively the vertices and edges reviewed above. A vertex $(\widehat{u}, X)$ is not implemented "directly" as a pair including the list X (because this would be time and space consuming) but by pointers towards $\widehat{u}$ and the sequence $N_{d(u)}(X)$ of length at most ∂. We order the sets $N_{d(u)}(X)$ according to $\leq_T$ in order to facilitate the comparison of two of them. We may have $N_{d(u)}(X) = \emptyset$. We will say that $N_{d(u)}(X)$ *encodes* X at node $\widehat{u}$ of T'.

Processing a node u of T is done as follows:

(a) If u is leaf of T, we add to the current graph (that contains T' and other vertices and edges) the vertex $(u, \{u\})$ and the edge $u \to (u, \{u\})$ of type (i).

(b) If u has sons $u_1, ..., u_p$ in T that have already been processed:

(b.1) we add to the current graph the vertices $(u, \{u\}), (\widehat{u}, \{u\})$ and $(\widehat{u}, Y)$ for sets Y defined below in (b.3),

(b.2) we add the edges of type (i) $u \to (u, \{u\})$, $\widehat{u} \to (\widehat{u}, \{u\})$ and the edges $\widehat{u} \to (\widehat{u}, Y)$ for all vertices $(\widehat{u}, Y)$ added by clause (b.3),

(b.3) we add the following edges of type (ii) :

(b.3.1) $(\widehat{u}, \{u\}) \to (u, \{u\})$,

(b.3.2) for each $i = 1, ..., p$, the edge $(\widehat{u}, Y) \to (u_i, \{u_i\})$ if u_i is a leaf of T and similarly, $(\widehat{u}, Y) \to (\widehat{u_i}, \{u_i\})$ if u_i is an inner node, where in both cases, the set Y is represented by $N_{d(u_i)-1}(u_i) = N_{d(u)}(Y)$; the set $N_{d(u_i)-1}(u_i)$ is given in the data structure for G,

(b.3.3) if u_i is an inner node, we add the edges $(\widehat{u}, Y) \to (\widehat{u_i}, X)$ such that Y is represented by $N_{d(u_i)}(X) - \{u_i\}$ if $(\widehat{u_i}, X)$ is in the current graph,

(b.4) we add the edges of type (iii) $(\widehat{u}, \{u\}) \to (\widehat{u}, Y)$ and $(\widehat{u}, Y) \to (\widehat{u}, \{u\})$ if $u \in N_{d(u)}(Y)$; (we add pairs of opposite edges because G is undirected).

Let us insist that the sets X and Y are not manipulated directly, but encoded and compared through the (in most cases) smaller sets $N_{d(u_i)}(X)$ and $N_{d(u)}(Y)$. The vertices $(\widehat{u}, Y)$ to be added by clause (b.1) are determined as the tails of the edges of type (ii) added by clauses (b.3.2) and (b.3.3).

Claim : The graph D constructed in this way is isomorphic to $D(T', \varpi)$.

Proof : We prove that each vertex $(\widehat{u}, X)$ of $D(T', \varpi)$ is added (and encoded as explained).

Let $x \in X$. It is a leaf of T. Let $x, u_1, ..., u_{m-1}, u$ be the ascending path in T. It is clear from (b.3.2) and (b.3.3) (by using an induction on $i \leq m$) that there is a unique sequence $X_1, ..., X_{m-1}$ such that $x \in X_1 \subseteq ... \subseteq X_{m-1} \subseteq X$ and $X_i \in \Omega(T_<(u_i))$ for each i, $X \in \Omega(T_<(u))$ and $(x, \{x\})$, $(\widehat{u_1}, X_1)$,..., $(\widehat{u_i}, X_i)$,..., $(\widehat{u}, X)$ are in D. By using Fact 3, we can avoid inserting in D two vertices representing a same vertex $(\widehat{u}, X)$ of $D(T', \varpi)$. The incident edges are inserted by clauses (b.3.2), (b.3.3) and (b.4).

All other vertices and edges of D are as in the definition of $D(T', \varpi)$.$\square$

Computation time.

We denote by n the number of vertices of the graph G, identical to the number of nodes of T, and by p the width of ϖ. The construction of T' takes time $O(n)$. We now bound the time taken to implement clause (b.3.3). For each son u_i of an inner node u, for each vertex $(\widehat{u_i}, X)$ created by processing u_i, we compute $W := N_{d(u_i)}(X) - \{u_i\}$, which is done in constant time and we check whether $W = N_{d(u)}(Y)$ for some existing vertex $(\widehat{u}, Y)$. If this is the case, we create the edge of type (ii) : $(\widehat{u}, Y) \rightarrow (\widehat{u_i}, X)$, otherwise, we do the same after having created $(\widehat{u}, Y)$ as a new vertex with the edge of type (i) : $\widehat{u} \rightarrow (\widehat{u}, Y)$. This takes time $O(p.\partial)$ for each $(\widehat{u_i}, X)$, hence total time $O(p^2.\partial.n)$ for all vertices $(\widehat{u_i}, X)$.

For clause (b.4), we do as follows: each time we add a vertex $(\widehat{u}, Y)$, we check if $u \in N_{d(u)}(Y)$ which can be done in constant time as u is the first element if it belongs to $N_{d(u)}(Y)$. The total time is thus $O(p.n)$.

All other steps take total time $O(n)$. Hence, $D = D(T', \varpi)$ is computed in time $O(p^2.\partial.n)$.

The algorithm for directed graphs.

If G is directed, the algorithm is similar except that we use, instead of $N_i(x)$, the pair of sets $(N_i^+(x), N_i^-(x))$ where $N_i^+(x) := \{y \in V_G \mid x <_T y, (x, y) \in edg_G, d(y) \leq i\}$ and $N_i^-(x) := \{y \in V_G \mid x <_T y, (y, x) \in edg_G, d(y) \leq i\}$.

The vertices $(\widehat{u}, X)$ of D are then encoded by pointers to $\widehat{u}$ and to the sets $N^+_{d(u)}(X)$ and $N^-_{d(u)}(X)$. The edges type (iii) to be added by clause (b.4) are :

$(\widehat{u}, \{u\}) \rightarrow (\widehat{u}, X)$ if $u \in N^-_{d(u)}(X)$ and $(\widehat{u}, X) \rightarrow (\widehat{u}, \{u\})$ if $u \in N^+_{d(u)}(X)$.

The computation time is also $O(p^2.\partial.n)$. $\square$

Theorem 19 : Let G be given by a normal tree T and the sets $N_{d(x)-1}(x)$. Let (T', ϖ) be the partition tree defined by Proposition 7. Let p be its width

and $\partial := \max\{\left|N_{d(x)-1}(x)\right| \mid x \in N_T\}$. A clique-width term of width p that denotes G can be computed in time $O(p^2.\partial.n)$. Its reduced term is T'.

Proof: Algorithm 18 takes time $O(p^2.\partial.n)$ to compute a graph D isomorphic to $D(T', \varpi)$. Then we can obtain the corresponding clique-width term in time $O(p^2.n)$ by a suitable adaptation of the algorithm of [8]. □

Remarks 20 : (1) In Algorithm 18, for G undirected, one can replace $N_i(x)$ by $D_i(x) := \{d(y) \mid y \in N_i(x)\}$ because if $x, y <_T u$, then $N_{d(u)}(x) = N_{d(u)}(y)$ if and only if $D_{d(u)}(x) = D_{d(u)}(y)$. The graph G is completely determined by T and, for each vertex x, the set $D_{d(x)-1}(x)$.

(2) We observe the following without giving the detailed proof. The relative clique-width of G with respect to T', denoted by $rcwd(G, T')$, is either p or $p+1$. It is p if and only if, for every node u of T such that $|\Omega(T_<(u))| = p$, the part $\{u\}$ of $\varpi(\widehat{u})$ can be merged with some part X of $\Omega(T_<(u))$. This is the case if u is not a neighbour of any vertex in $T_<(u)$ and if, unless u is the root, the set $\{u\} \cup X$ is included in some part of $\Omega(T_<(p_T(u)))$. It follows that an easy modification of Algorithm 18 yields a clique-width term of width $rcwd(G, T')$, hence, an optimal term for the relative clique-width of G with respect to T'.

Determining if the relative clique-width of a graph with respect to a given tree is at most some given number is an NP-complete problem [19], but here the given tree is not arbitrary because it is based on a normal tree. This explains why we can obtain a polynomial time algorithm.

(3) The construction of Theorem 15 can be translated similarly into an algorithm that builds a clique-term defining $Bip(H)$ of width $O(twd(H)^{q-1})$ in time $O(twd(H)^{2q-1}.n)$. This time bound is based on Theorem 19.

(4) The terms constructed by Theorem 19 use the operations $\overrightarrow{add}_{a,b}$ and $add_{a,b}$ in a restricted way: these operations are applied to graphs having at most one a-labelled vertex or at most one b-labelled vertex. Hence, we cannot hope to obtain in this way terms that witness clique-width in all cases.

6 Conclusion

For uniformly m-sparse graphs, clique-width is polynomially bounded in terms of tree-width and we have algorithmically efficient transformations of 1-downwards increasing tree-decompositions into clique-width terms witnessing the claimed upper-bounds.

Our proofs and the corresponding algorithms use normal trees as data structures for tree-decompositions. Furthermore, we have used the characterization of clique-width of [8] based on partition trees, and the associated graph representations. We think that these tools yield clear proofs and efficient algorithms.

The applications to FPT graph algorithms for checking monadic second-order properties are developped in [4].

We propose two open questions :

Does there exist $p < m$ such that $cwd(G) = O(twd(G)^p)$ for every uniformly m-sparse graph G ?

Can one transform efficiently a tree-decomposition of width k of a graph G of degree at most d into a clique-width term denoting G of width $O(d.k)$ (cf. Theorem 6) ?

References

[1] T. Bouvier, *Graphes et décompositions*, Doctoral dissertation, Bordeaux University, 2014.

[2] D. Corneil and U. Rotics, On the relationship between clique-width and tree-width. *SIAM J. Comput.* **34** (2005) 825-847.

[3] B. Courcelle, On the model-checking of monadic second-order formulas with edge set quantifications, *Discrete Applied Mathematics* **160** (2012) 866-887.

[4] B. Courcelle, Fly-automata for checking monadic second-order properties of graphs of bounded tree-width, *Proceedings of LAGOS 2015*, Beberibe, Brazil, to appear in *Electronic Notes in Discrete Mathematics*; a long version is in http://www.labri.fr/perso/courcell/Conferences/BC-Lagos2015.pdf

[5] B. Courcelle and I. Durand, Automata for the verification of monadic second-order graph properties, *J. Applied Logic* **10** (2012) 368-409.

[6] B. Courcelle and I. Durand, Computations by fly-automata beyond monadic second-order logic, June 2013, http://hal.archives-ouvertes.fr/hal-00828211, to appear in *Theor. Comput. Sci*, Short version in *Proc. Conference on Algebraic Informatics*, *Lecture Notes in Computer Science* **8080** (2013) 211-222.

[7] B. Courcelle and J. Engelfriet, *Graph structure and monadic second-order logic, a language theoretic approach*, Volume **138** of *Encyclopedia of mathematics and its application*, Cambridge University Press, June 2012.

[8] B. Courcelle, P. Heggernes, D. Meister, C. Papadopoulos and U. Rotics, A characterisation of clique-width through nested partitions, *Discrete Applied Maths*, **187** (2015) 70-81.

[9] B. Courcelle, J. Makowsky and U. Rotics, Linear-time solvable optimization problems on graphs of bounded clique-width, *Theory Comput. Syst.* **33** (2000) 125-150.

[10] R. Diestel, *Graph theory*, Third edition, Springer, 2006.

[11] R. Downey and M. Fellows, *Parameterized complexity*, Springer-Verlag, 1999.

[12] R. Downey and M. Fellows, *Fundamentals of parameterized complexity*, Springer-Verlag, 2013.

[13] F. Gurski and E.Wanke, The tree-width of clique-width bounded graphs without $K_{n,n}$. *Proceedings of 26th Workshop on Graphs (WG)*, *Lecture Notes in Computer Science* **1928** (2000) pp. 196-205.

[14] E. Fischer J. Makowsky and E. Ravve, Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics* **156** (2008) 511-529.

[15] F. Fomin, S. Oum, D. Thilikos, Rank-width and tree-width of H-minor-free graphs. *Eur. J. Comb.* **31** (2010) 1617-1628.

[16] A. Frank, Connectivity and networks, in: *Handbook of Combinatorics, Vol.1*, Elsevier 1997, pp. 111-178.

[17] M. Kanté and M.Rao, The rank-width of edge-coloured graphs. *Theory Comput. Syst.* **52** (2013) 599-644.

[18] V. Lozin and D. Rautenbach, The relative clique-width of a graph. *J. Comb. Theory, Ser. B* **97** (2007) 846-858.

[19] H. Müller and R. Urner, On a disparity between relative clique-width and relative NLC-width, *Discrete Applied Mathematics* **158** (2010) 828-840.

[20] S. Oum, Rank-width is less than or equal to branch-width. *Journal of Graph Theory* **57** (2008) 239-244.

[21] S. Oum, Rank-width and vertex-minors. *J. Comb. Theory, Ser. B* **95** (2005) 79-100.

[22] S. Oum, P. Seymour, Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B* **96** (2006) 514-528.

[23] D. Wood, On tree-partition-width, *European Journal of Combinatorics* **30** (2009) 1245–1253.